ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

Διπλωματική Εργασία

Σχεδιασμός και υλοποίηση μεθόδων επιλογής σημείου πρόσβασης σε ασύρματα δίκτυα με τη χρήση πειραματικών διατάξεων ανοιχτού λογισμικού

Μπαράτι – Ντεχντεζι Χαμιντ (Νικόλας)

Επιβλέποντες

Κουτσόπουλος Ιορδάνης, Επίκουρος Καθηγητής
Τασιούλας Λέανδρος, Καθηγητής

10

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

Διπλωματική Εργασία

Σχεδιασμός και υλοποίηση μεθόδων επιλογής σημείου πρόσβασης σε ασύρματα δίκτυα με τη χρήση πειραματικών διατάξεων ανοιχτού λογισμικού

Μπαράτι – Ντεχντεζι Χαμιντ (Νικόλας)

Επιβλέποντες

Κουτσόπουλος Ιορδάνης, Επίκουρος Καθηγητής
Τασιούλας Λέανδρος, Καθηγητής

10

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

Διπλωματική Εργασία

Σχεδιασμός και υλοποίηση μεθόδων επιλογής σημείου πρόσβασης σε ασύρματα δίκτυα με τη χρήση πειραματικών διατάξεων ανοιχτού λογισμικού

Μπαράτι – Ντεχντεζι Χαμιντ (Νικόλας)

Εγκρίθηκε από την διμελή εξεταστική επιτροπή τον Οκτώβριο του 2010.

.....	**.....**
Ι. Κουτσόπουλος	**Λ.Τασιούλας**
Επίκουρος Καθηγητής	**Καθηγητής**

Διπλωματική εργασία για την απόκτηση του διπλώματος του Μηχανικού Ηλεκτρονικών Υπολογιστών, Τηλεπικοινωνιών και Δικτύων του Πανεπιστημίου Θεσσαλίας, στα πλαίσια του Προγράμματος προπτυχιακών σπουδών του Τμήματος Μηχανικών Η/Υ, Τηλεπικοινωνιών και Δικτύων του Πανεπιστημίου Θεσσαλίας.

.....

Μπαρατι – Ντεχντεζι Χαμιντ (Νικόλας)
Διπλωματούχος Μηχανικός Ηλεκτρονικών Υπολογιστών, Τηλεπικοινωνιών και Δικτύων.

Στον Πατέρα μου Ρεζά

Ευχαριστίες:

Η παρούσα διπλωματική εργασία δε θα μπορούσε ποτέ να ολοκληρωθεί χωρίς την πολύτιμη βοήθεια ανθρώπων που σ' όλη τη διάρκεια αυτών των μηνών ενασχόλησης και έρευνας μου προσέφεραν απλόχερα τις γνώσεις και τη βοήθειά τους και γι 'αυτό νιώθω την ανάγκη να τους ευχαριστήσω ειλικρινά γι όλη την υποστήριξη που μου παρείχαν.

Ευχαριστώ θερμά τον καθηγητή κύριο Αθανάσιο Κοράκη, ο οποίος με τις υποδείξεις, τις ιδέες και την καθοδήγησή του άνοιξε το μεγαλύτερο μέρος του δρόμου που οδήγησε εδώ στην περάτωση της εργασίας αυτής.

Ευχαριστώ πολύ τους επιβλέποντες καθηγητές μου κυρίους Κουτσόπουλο και Τασιούλα που δέχτηκαν να αναλάβουν την επίβλεψη της διπλωματικής μου.

Ευχαριστώ τον διδακτορικό φοιτητή Στράτο Κερανίδη, που με τις συμβουλές και τις γνώσεις του σχετικά με τον Madwifi driver και το Testbed βοήθησε με καθοριστικό τρόπο να προχωρήσω και να αντιμετωπίσω τυχόν δυσκολίες που παρουσιάζονταν κάθε τόσο.

Ευχαριστώ επίσης τους Απόστολο Αποστολάρα, Νίκο Γιαλελή και την υπόλοιπη ομάδα του Nitlab καθώς και τους συμφοιτητές μου, Γιώργο Κυριακού και Δονάτο Σταυρόπουλο για την βοήθεια και τη συνδρομή τους στην όλη προσπάθειά μου.

Τέλος, ευχαριστώ θερμά τον συμφοιτητή μου Νικόλαο Μακρή ο οποίος με την εμπειρία του, τις ιδέες, τη συμπαράσταση ακόμα και με προσφορά υλικού με διευκόλυνε πολύ στην περάτωση της παρούσας εργασίας.

Μπαράτι Νικόλας
Βόλος, 2010

Περιεχόμενα

ΚΕΦΑΛΑΙΟ 1

Εισαγωγή:

Έχουν περάσει πλέον αρκετά χρόνια από την είσοδο των ασύρματων δικτύων στην καθημερινότητά μας. Τα προφανή πλεονεκτήματά τους, κυρίως το χαμηλό κόστος εγκατάστασης λόγω απουσίας μεγάλης υποδομής, εκτόξευσαν τη χρήση και την δημοτικότητά τους και εδραίωσαν την παρουσία τους. Από την άλλη τα , προφανή επίσης, προβλήματα που συν-ακολουθούν κάθε καινοτομία και τα μεγάλα περιθώρια βελτιστοποίησης που αφήνει η νέα αυτή τεχνολογία κέντρισαν το ενδιαφέρον των ερευνητών σ' όλον τον κόσμο.
Το κύριο πεδίο έρευνας για όλους ήταν και είναι η βελτιστοποίηση της απόδοσης των δικτύων αυτών. Κύριοι δείκτες μέτρησης αυτής της απόδοσης είναι το συνολικό throughput του συστήματος δηλαδή ο ρυθμός επιτυχημένης παράδοσης πακέτων στο κανάλι επικοινωνίας, και η κατανάλωση ενέργειας. Στόχος πάντα είναι μεγιστοποίηση του πρώτου και η ελαχιστοποίηση του δεύτερου ειδικά σε ενεργειακά κρίσιμα πεδία εφαρμογής όπως δίκτυα sensors. Όσον αφορά τη μεγιστοποίηση του throughput, προκύπτει μια σειρά από ζητήματα η λύση των οποίων είναι προϋπόθεση για την επίτευξη του στόχου αυτού. Ζητήματα όπως διαμοιρασμός του καναλιού, rate selection/adaptation, seameless roaming, γρήγορο scanning και επιλογή κατάλληλου σημείου πρόσβασης (**A**ccess **P**oint).

Στην εργασία αυτή ασχολούμαστε με 2 από τα παραπάνω θέματα. Θα χρησιμοποιήσουμε δεδομένα που παράγονται από τον μηχανισμό διαμοιρασμού του καναλιού που χρησιμοποιούν τα ασύρματα δίκτυα που υπακούν στο πρότυπο του IEEE80211 με σκοπό να δημιουργήσουμε μια αξιόπιστη μετρική βάσει της οποίας οι σταθμοί θα επιλέγουν τα πλέον κατάλληλα AP, εκείνα δηλαδή που εξασφαλίζουν τόσο για το σταθμό όσο για το σύστημα συνολικά, μεγαλύτερο throughput.

Στόχοι της Παρούσας Εργασίας

Η παρούσα εργασία έχει ως στόχο την παρουσίαση ενός νέου σχήματος association σταθμών σε AP εισάγοντας μια νέα μετρική η οποία χρησιμοποιεί δεδομένα που προκύπτουν από τη διαδικασία του Backoff. Η τελευταία είναι η διαδικασία με την οποία ένα IEEE80211 δίκτυο “διαμοιράζει το κανάλι” και προσπαθεί να αποφύγει συγκρούσεις και επικαλύψεις μεταξύ των σταθμών που επιθυμούν να στείλουν δεδομένα. Είναι ένα πλήρως κατανεμημένο σχήμα το οποίο θα αναλυθεί στη συνέχεια όπου θα φανεί καλύτερα ο ρόλος του στην εξασφάλιση ομαλής και αποδοτικής λειτουργίας του δικτύου.
Στην κατεύθυνση του νέου αυτού σχήματος θα παρουσιαστεί ένας αλγόριθμος ο οποίος θα περιγράφει και θα ρυθμίζει τη λειτουργία αυτού του νέου σχήματος.
Να σημειωθεί ότι ως όχημα για την υλοποίηση όλων αυτών χρησιμοποιήθηκε ο driver ανοιχτού κώδικα Madwifi και όχι κάποιο πρόγραμμα προσομοίωσης, αποσκοπώντας στην επίτευξη δημιουργίας όσο το δυνατόν πιο ρεαλιστικών συνθηκών λειτουργίας και δοκιμής.
Τα παραπάνω θα δοκιμαστούν ως προς τα όσα υπόσχονται στις πειραματικές διατάξεις (Testbed) του Nitlab του τμήματος Μηχανικών Η/Υ Τηλεπικοινωνιών και Δικτύων του Πανεπιστημίου Θεσσαλίας.

Διάρθρωση της Εργασίας

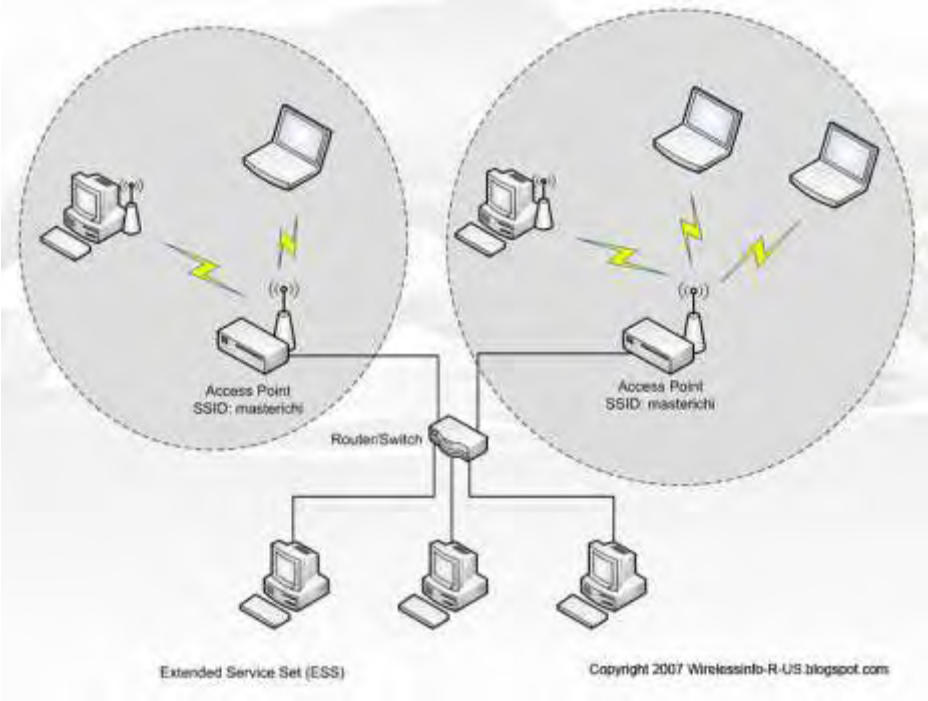
Η συνέχεια της εργασίας έχει ως εξής:

- Παρουσίαση της διαδικασίας του association στα IEEE80211 δίκτυα
- Περιγραφή του Backoff
- Σύντομη παρουσίαση του NITOS Testbed
- Σύντομη περιγραφή του Madwifi
- Soft Backoff: Η Δική μου εργασία πάνω στο Backoff
- Το νέο association σχήμα
- Πειραματικά αποτελέσματα
- Επίλογος
- Παράρτημα με κομμάτια του κώδικα του Madwifi που υπέστησαν αλλαγές

ΚΕΦΑΛΑΙΟ 2

Association στο IEEE80211

Στο πρότυπο IEEE80211, κάθε σταθμός, πριν την έναρξη των διαδικασιών αποστολής και λήψης πακέτων δεδομένων, πρέπει να συνδεθεί σ' ένα Access Point το οποίο του εξασφαλίζει την επικοινωνία με άλλους σταθμούς εντός του δικτύου που συγκροτείται (BSS: Basic Service Set, εικόνα 2.1) από το AP και τους σταθμούς που έχουν συνδεθεί ήδη με αυτόν, ή με ένα μεγαλύτερο δίκτυο για το οποίο αποτελεί gateway το εν λόγω AP. Το τελευταίο ισχύει αν το AP έχει δυνατότητες σύνδεσης με τον "έξω κόσμο" ή αποτελεί κομμάτι ενός ESS (Extended Service Set). Όπως στην εικόνα 2.1.
Η όλη διαδικασία του association έχει 4 φάσεις. Οι φάσεις αυτές ξεκινούν με την έναρξη της λειτουργίας του σταθμού ή την είσοδο του σε νέα περιοχή όπου υπάρχουν Access Points.
Στην πρώτη φάση ο σταθμός προσπαθεί να ανιχνεύσει τα Access Points γύρω του για να συνδεθεί με κάποιον από αυτούς. Η φάση αυτή ονομάζεται scanning και στο τέλος αυτής ο σταθμός θα έχει στη διάθεσή της μια λίστα από υποψήφιους προς σύνδεση AP.



Εικόνα 2.1

Το πρότυπο IEEE80211 καθορίζει τους δυο παρακάτω τρόπους scanning.

- passive
- active

Στον πρώτο ο σταθμός σαρώνει όλες τις διαθέσιμες συχνότητες που καθορίζει το πρωτόκολλο στο οποίο υπακούει (π. χ. στον IEEE80211g από 2412 μέχρι 2483 GHz) και συλλέγει παθητικά ειδικά broadcast πακέτα που στέλνουν τα AP στην εμβέλεια των οποίων βρίσκεται και ονομάζονται Beacons. Αυτά τα ειδικά πακέτα περιέχουν τις απαραίτητες πληροφορίες σχετικά με το AP που τα έστειλε. οι πληροφορίες αυτές είναι:

- Χρονοσφραγίδα

  Το πότε στάλθηκε το παρόν beacon
- Beacon Interval

Το χρονικό διάστημα ανάμεσα στην αποστολή 2 διαδοχικών beacons. Επιδέχεται τροποποίηση στον AP και μετριέται σε Time Units και είναι συνήθως ίσο με 110 milisecond.

- Capability Informations

Είναι ένα πεδίο 16 bit και περιέχει πληροφορίες για το είδος / δυνατότητες του δικτύου, π. χ. αν το δίκτυο είναι ad-hoc ή όχι, αν υποστηρίζει polling ή κρυπτογράφηση κλπ.

- Το αναγνωριστικό (ID) του δικτύου
- Το σύνολο των ρυθμών μετάδοσης που υποστηρίζει
- Και τέλος ένα σύνολο από παραμέτρους που αφορούν το Frequency Hopping, Direct Sequencing, Contention Free, και IBSS.

Στο active scanning, ο σταθμός είναι που παίρνει την πρωτοβουλία και στέλνει ειδικά broadcast πακέτα ονόματι Probe Request σε όλες τις συχνότητες του πρωτοκόλλου στο οποίο υπακούει. Αυτά τα ειδικά πακέτα περιέχουν το αναγνωριστικό του - υποψήφιου προς σύνδεση - δικτύου και το σύνολο των ρυθμών μετάδοσης που υποστηρίζει ο σταθμός.
Η αποστολή των Probe Request στα AP του περιβάλλοντος του σταθμού προκαλούν την αποστολή άλλων ειδικών πακέτων από τη μεριά των APs που ονομάζονται Probe Responses και έχουν πανομοιότυπο περιεχόμενο με τα Beacons όπως αναλύθηκε πιο πάνω.

Μετά το τέλος της φάσης του scanning ο σταθμός έχει στη διάθεσή του μια λίστα από APs και μπορεί να προχωρήσει στην επιλογή ενός από αυτά. Να σημειωθεί εδώ ότι το IEEE80211 επιτρέπει τη σύνδεση ενός σταθμού μόνο σε ένα AP.
Το κριτήριο επιλογής, όπως επιτάσσει το πρότυπο IEEE80211 είναι το RSSI (Received Singnal Strength Indicator) ή αλλιώς Δείκτης Ισχύος Λαμβανόμενου Σήματος. Η επιλογή Σημείου Πρόσβασης βάσει αυτής της μετρικής έχει αναδείξει μερικά ενδιαφέροντα προβλήματα τα οποία δείχνουν ότι ένα AP με μεγάλο RSSI δεν είναι πάντα η καλύτερη των επιλογών μιας και αυτή η μετρική λαμβάνει υπόψιν της μόνο το καθοδικό κανάλι (Down Link) και αγνοεί παραμέτρους όπως φόρτος του δικτύου, θόρυβος γύρω από το AP, θόρυβος γύρω από τον σταθμό, και άλλα. Ένα μεγάλο μέρος ερευνών που γίνονται στον χώρο των ασυρμάτων

δικτύων αφορούν την εύρεση νέων μετρικών που θα μπορέσουν να αντικαταστήσουν το RSSI. Και η παρούσα διπλωματική αποτελεί ένα μικρό μέρος αυτών.

Στην τρίτη φάση, εφόσον ο σταθμός επέλεξε ένα Σημείο Πρόσβασης στο οποίο επιθυμεί να συνδεθεί, βρίσκεται η διαδικασία της αυθεντικοποίησης. Ανάλογα με το mode αυθεντικοποίησης που υποστηρίζει το AP, WEP, WPA, WPA2 , ο σταθμός θα πρέπει να “πείσει” ότι ανήκει στην ομάδα των σταθμών εκείνων που εμπιστεύεται ο διαχειριστής του δικτύου να απολαμβάνουν της υπηρεσίες του AP. Αυτή η διαδικασία, εφόσον είναι ενεργοποιημένη στον AP, περιλαμβάνει μια αποστολή username και password από τον σταθμό και επιβεβαίωση ή άρνηση ανάλογα με το αν το ζεύγος username - password είναι σωστά, από το AP.

Τέλος, στην τέταρτη φάση βρίσκεται τελική απάντηση του AP. Αυτή, δίνεται όταν σταλεί από τον σταθμό αίτημα σύνδεσης (Association Request). Αυτό το αίτημα μεταξύ άλλων περιλαμβάνει:

- Capability Informations,
  Όπως εξηγήθηκε πιο πάνω
- Αναγνωριστικό του δικτύου (SSID)
- Υποστηριζόμενοι ρυθμοί μετάδοσης

Η αποστολή του αιτήματος αυτού προκαλεί την απάντηση του AP σε μορφή ειδικού πακέτου που ονομάζεται Association Response και περιλαμβάνει μεταξύ άλλων τα εξής πεδία:

- Capability Informations
- Status Code
  Εδώ ουσιαστικά βρίσκεται η απάντηση του Access Point. Το AP μπορεί να αρνηθεί τη σύνδεση του σταθμού για διάφορους λόγους. π. χ λόγω υπερβολικού φόρτου, δηλαδή όταν προσθήκη ενός νέου σταθμού θα ξεπεράσει το άνω όριο επιτρεπόμενων συνδεδεμένων σταθμών. Αυτό το όριο στο Madwifi είναι 120.
- Αναγνωριστικό Association (AID)
- Υποστηριζόμενοι ρυθμοί μετάδοσης

Εικόνα 2.2

Στην παραπάνω εικόνα φαίνονται οι τέσσερεις φάσες που μόλις περιγράφτηκαν.

ΚΕΦΑΛΑΙΟ 3

Πρόσβαση στο Κανάλι και Backoff

Αφού λοιπόν συνδέθηκε ο σταθμός σε κάποιο σημείο πρόσβασης που επέλεξε, είναι πλέον σε θέση να δέχεται και να στέλνει πακέτα πληροφοριών από και προς το δίκτυο. Ας υποθέσουμε ότι το δίκτυό μας έχει την παρακάτω μορφή. Όπου οι 2, οι 3 ή ακόμα και οι 4 σταθμοί θέλουν να στείλουν δεδομένα στο AP.

Εικόνα 3.1

Αν αποφασίσουν να το πράξουν ταυτόχρονα, ή, αν όχι απόλυτα συγχρονισμένα αλλά ενόσω κάποια μετάδοση είναι ήδη σε εξέλιξη, τα πακέτα που μεταδίδονται δε θα φτάσουν ποτέ στον προορισμό τους ή θα φτάσουν κατεστραμμένα και αδύνατα να αναγνωστούν.
Προς αντιμετώπιση αυτού του προβλήματος, του πως δηλαδή θα διαμοιράζεται το κοινό μέσον διάδοσης ώστε να επιτευχθεί μια αποδοτική επικοινωνία μεταξύ των μελών ενός δικτύου έχουν προταθεί κατά καιρούς διάφορα σχήματα. Μερικά εισήγαγαν χρονική διαμοίραση στο κανάλι αναθέτοντας συγκεκριμένες χρόνο-θυρίδες κατά τη διάρκεια των οποίων κάθε σταθμός μπορεί να μεταδίδει τα πακέτα του. Άλλα, διαμοίραζαν το κανάλι σε συχνότητες. Άλλα, όπως το πρωτόκολλο pure Aloha δεν εισήγαγαν κανένα περιορισμό και απλά επέτρεπαν την μετάδοση πακέτων όποτε ο κάθε σταθμός επιθυμούσε. Αν η μετάδοση αποτύγχανε ο σταθμός δεν είχε παρά να ξανά προσπαθήσει να στείλει το πακέτο. Ένα πολύ απλό σχήμα το οποίο οδηγούσε σε πολύ φτωχή απόδοση.

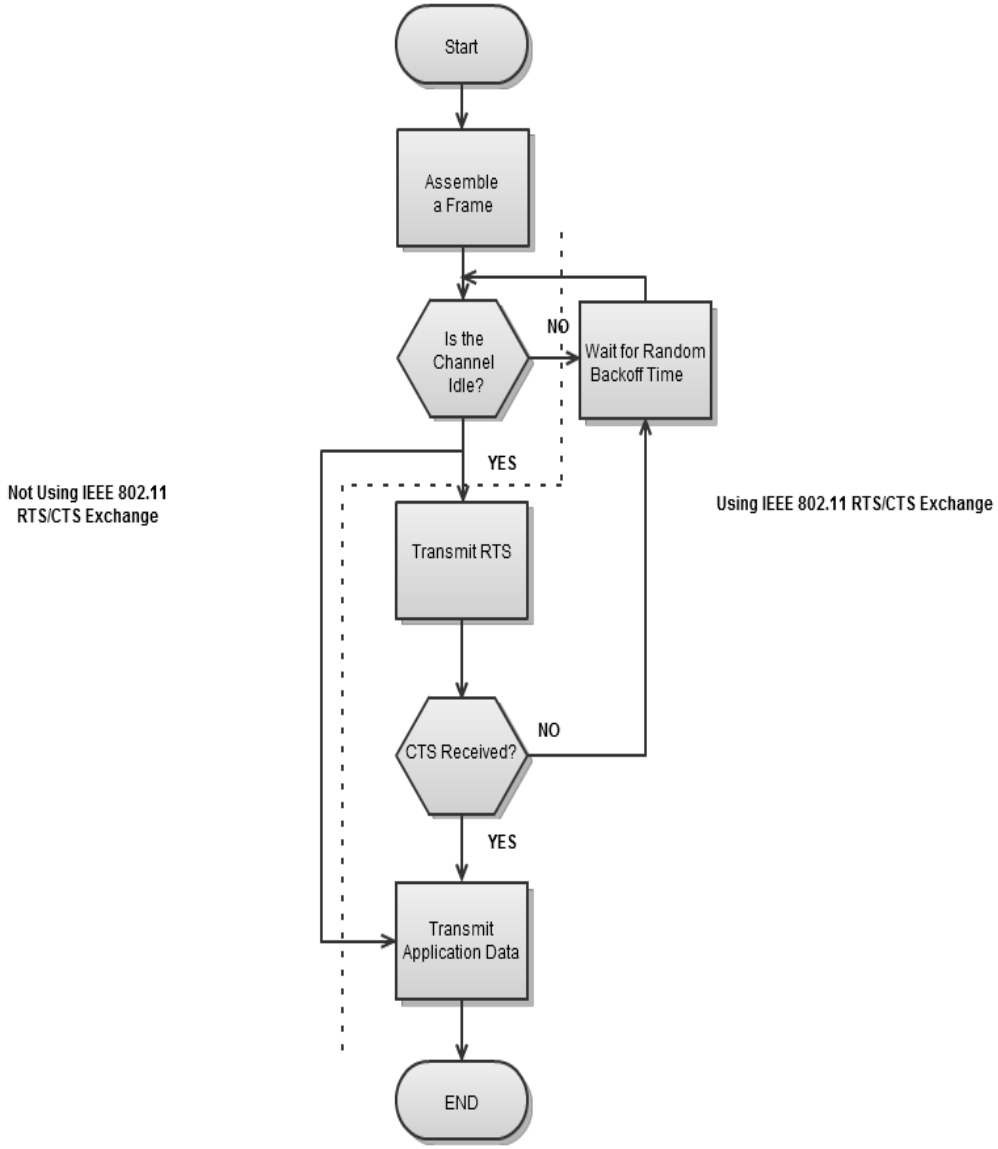
Σήμερα για την επίτευξη υψηλότερων αποδόσεων σε (ασύρματα) δίκτυα υπολογιστών, έχει καθιερωθεί ένας πιθανοτικός μηχανισμός με σκοπό την ελαχιστοποίηση των συγκρούσεων των πακέτων και την αλληλο-επικάλυψη των μεταδόσεων. Ο μηχανισμός αυτός ονομάζεται Carrier Sense Multiple Access (CSMA) ή στα ελληνικά Ανίχνευση Μέσου Πολλαπλής Πρόσβασης και ενώ δεν λύνει οριστικά και ντετερμινιστικά το πρόβλημα, μειώνει όμως σημαντικά την πιθανότητα συγκρούσεων.
Σύμφωνα με το παραπάνω μηχανισμό, ο κάθε σταθμός πριν αποπειραθεί να στείλει κάποιο πακέτο, “ακούει” το μέσον μετάδοσης για κάποιο χρονικό διάστημα και αν δεν ανιχνεύσει κάποια άλλη παράλληλη μετάδοση ξεκινά τη μετάδοσή του. Όπως φαίνεται, το CSMA δεν εξαλείφει πλήρως το πρόβλημα, μιας και 2 σταθμοί για παράδειγμα μπορεί να ξεκινήσουν τη μετάδοσή τους ταυτόχρονα αφού έτυχε να ακούσουν το κανάλι ανενεργό για το ίδιο χρονικό διάστημα.

Το CSMA έχει 2 εκδοχές οι οποίες θα αναλυθούν παρακάτω μαζί με τον αλγόριθμο που καθορίζει τη λειτουργία τους. ΟΙ 2 εκδοχές είναι:

- CSMA/CA
- CASMA/CM

Στην πρώτη εκδοχή τα αρχικά CA αναφέρονται στις λέξεις Collision Avoidance (Αποφυγή Συγκρούσεων). Χρησιμοποιείται στα ασύρματα δίκτυα και είναι μια τροποποίηση του CSMA, που προσπαθεί να μειώσει ακόμα περισσότερο την πιθανότητα συγκρούσεων των πακέτων. Και εδώ, ο σταθμός που επιθυμεί να στείλει κάποιο πακέτο, ακούει το κανάλι για ένα διάστημα DIFS (DCF Inter-Farme Spacing), αν σ' αυτό το διάστημα δεν ανιχνευτεί κάποια κίνηση στο κανάλι, ο σταθμός ξεκινά τη μετάδοση, αλλιώς περιμένει. Σε περίπτωση που συμβεί κάποια σύγκρουση ή αλληλο-αποκαλυφθούν δυο μεταδόσεις, ο κάθε σταθμός περιμένει για κάποιο τυχαίο χρονικό διάστημα και στο τέλος αυτού του διαστήματος στέλνει το πακέτο. Το τυχαίο αυτό διάστημα επιλέγεται σε ένα μεταβαλλόμενο εύρος αριθμών, αρχικά από 1 έως 15, και η διαδικασία επιλογής αυτού του διαστήματος ονομάζεται Exponential Backoff και θα επεξηγηθεί αναλυτικά παρακάτω.
Η όλη διαδικασία μπορεί να εμπλουτιστεί ακόμα περισσότερο με προσθήκη μιας ανταλλαγής μηνυμάτων μεταξύ του αποστολέα και του δέκτη όπου ο πρώτος στέλνει ένα ειδικό μήνυμα RTS (Request To Send) ζητώντας να μάθει αν ο δέκτης είναι σε θέση να λάβει το πακέτο που πρόκειται να σταλεί. Αν όλα είναι εντάξει και ο δέκτης είναι σε θέση να λάβει το πακέτο στέλνει ως απάντηση στο RTS ένα CTS (Clear To Send) πακέτο και έτσι ξεκινά η μετάδοση. Να σημειωθεί εδώ ότι το CSMA/CA χωρίς την προσθήκη των RTS/CTS παίρνει υπ' όψιν μόνο το περιβάλλον του πομπού αγνοώντας αν υπάρχει κάποιος θόρυβος - παράλληλη μετάδοση στην πλευρά του δέκτη που θα τον εμπόδιζε να δεχτεί το πακέτο. Σε ένα σενάριο με 3 σταθμούς Α, Β, Γ όπου ο Α θέλει να στείλει δεδομένα στον Β και στο περιβάλλον του Β υπάρχει μια παράλληλη μετάδοση από τον Γ που όμως επειδή δεν είναι "ορατός" από τον Α αλλά δημιουργεί θόρυβο γύρω από τον Β, Ο Α θα πρέπει να περιμένει να λάβει CTS από τον Β για να μπορέσει να ξεκινήσει την μετάδοση. Αλλιώς τα δεδομένα που στέλνει θα συγκρουστούν με τα πακέτα που μεταδίδει ο Γ.

Το παραπάνω στη βιβλιογραφία ονομάζεται στη hidden terminal problem και είναι από τα πρώτα που παρουσιάστηκαν και αναλύθηκαν μετά την εφαρμογή του CSMA/CA στα ασύρματα δίκτυα.
Στην επόμενη εικόνα φαίνεται αλγοριθμικά η όλη λειτουργιά του CSMA/CA.



Εικόνα 3.2

Το CSMA/CD (Collision Detection) βρήκε εφαρμογή μόνο στα ενσύρματα δίκτυα και μάλιστα στις παλιές εκδόσεις του συζευγμένου (twisted pair) Ethernet όπου ο εντοπισμός συγκρούσεων ήταν εφικτός. Τα νέα (ενσύρματα πάντα) δίκτυα με

switches και full duplex συνδέσεις δεν χρησιμοποιούν πλέον την τεχνολογία αυτή και γι’ αυτό δεν θα αναλυθεί περαιτέρω.

Όπως αναφέρθηκε πιο πάνω, το CSMA/CA όταν παρουσιαστεί κάποια σύγκρουση προχωρεί στην επίλυση της, ή μάλλον προσπαθεί να αποφύγει μια νέα σύγκρουση χρησιμοποιώντας τον μηχανισμό του Exponential Backoff. Πριν επεξηγηθεί αναλυτικά το backoff, θα παρουσιαστούν κάποιες έννοιες απαραίτητες στην κατανόηση του μηχανισμού αυτού.

- DCF:

  Distributed Coordination Function

Πρόκειται για την τεχνική που εφαρμόζουν υποχρεωτικά όλα τα WLAN δίκτυα που υπακούν στο πρότυπο IEEE80211 και χρησιμοποιεί το CSMA/CA που περιγράφηκε ήδη και τον μηχανισμό του Backoff.

- SIFS:

  Short Inter Frame Spacing

Είναι το μικρότερο διάστημα και δίνει χρησιμοποιείται από πακέτα υψηλής προτεραιότητας όπως Acks και CTS πακέτα. Ισούται με περίπου 9 μs.

- DIFS:

  DCF Inter Frame Spacing

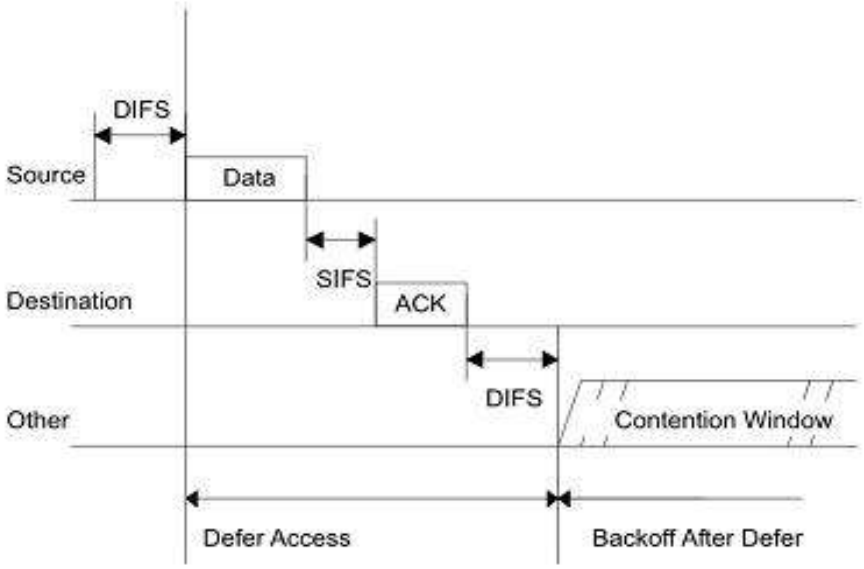
  Για τα υπόλοιπα πακέτα χαμηλότερης προτεραιότητας. Είναι το ελάχιστο χρονικό διάστημα που ένας σταθμός πρέπει να ακούσει το κανάλι ανενεργό πριν ξεκινήσει τη μετάδοση των δεδομένων του. Ισούται με SIFS συν 2 φορές το SLOT TIME το οποίο είναι 20μs.

- Contention Window:

Μεταβλητό, από τον μηχανισμό του backoff, χρονικό διάστημα κατά το οποίο ένας σταθμός περιμένει μέχρι να ξανά προσπαθήσει να μεταδώσει μετά από μια σύγκρουση. Συμβολίζεται με cw και η ελάχιστη δυνατή τιμή που μπορεί να πάρει

είναι 15 time slots και η μέγιστη 1023 time slots. CWmin και CWmax αντίστοιχα συμβολίζουν αυτές τις ελάχιστες και μέγιστες τιμές.

Στην επόμενη εικόνα φαίνονται σχηματικά οι σχέσεις μεταξύ των παραπάνω.



Εικόνα 3.3

Αλγόριθμος του Backoff:

Όπως ειπώθηκε πιο πάνω κάθε σταθμός ακούει το μέσο μετάδοσης για ένα χρονικό διάστημα DIFS. Μετά το πέρας αυτού του διαστήματος και αν το κανάλι ήταν άδειο, ή ανενεργό στέλνει τα δεδομένα του. Όταν παρουσιαστεί σύγκρουση, πράγμα που ερμηνεύεται από την απουσία Ack πακέτου, ξεκινά η διαδικασία του Backoff.
Σύμφωνα μ’ αυτήν λοιπόν,

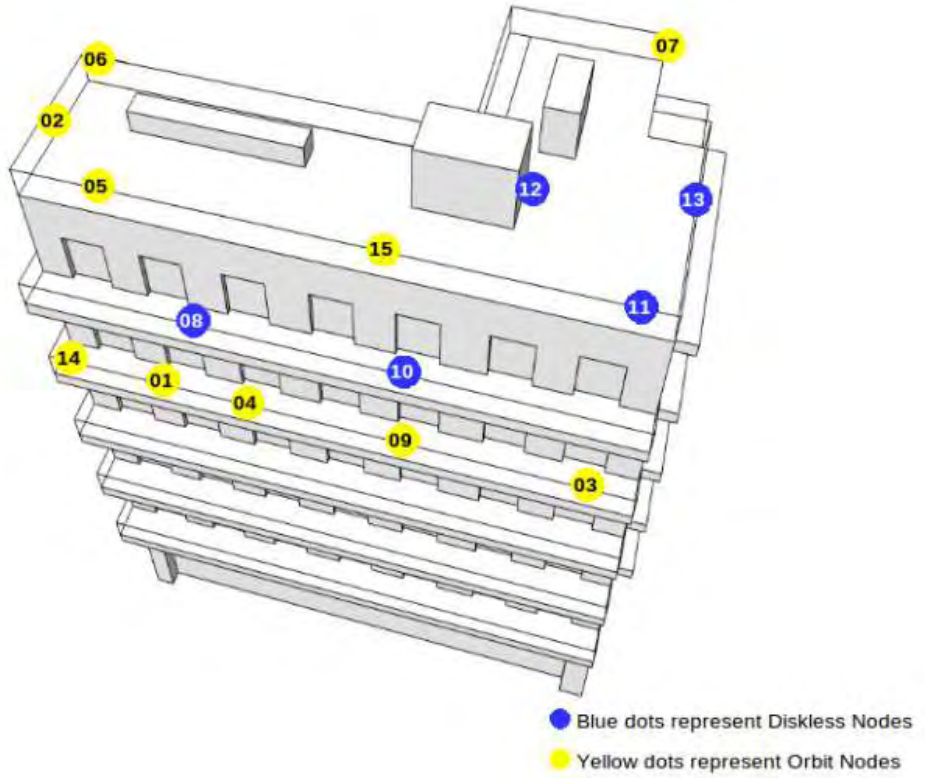
- επιλέγεται ένας τυχαίος αριθμός από μία ομοιόμορφη κατανομή σε ένα διάστημα μεταξύ CWmin και CWmax, αρχικά 15 και 31 αντίστοιχα. Αυτό το διάστημα όπως προαναφέρθηκε είναι το Contention Windowή cw.

- Κάθε time slot κατά το οποίο το κανάλι είναι ανενεργό, κάτι που διαπιστώνεται μέσω Carrier Sensing, ο σταθμός μετρά αντίστροφα.
- Όταν η μέτρηση φτάσει στο μηδέν, αποστέλλεται το πακέτο.
- Αν παρουσιαστεί πάλι σύγκρουση, το διάστημα από την οποία θα επιλεχθεί πάλι ένας τυχαίος αριθμός θα διπλασιαστεί. Για την ακρίβεια το cw θα έχει μέγεθος $min\,(2^i \times cwmin - 1, cwmax - 1)$
- Και πάλι γίνεται αντίστροφη μέτρηση και στον μηδενισμό ο σταθμός στέλνει τον πακέτο.

Μετά από κάθε επιτυχημένη αποστολή το διάστημα που ορίζει το cw γίνεται πάλι 15 με 31. Στις 6 διαδοχικές αποτυχίες, μιας και τόσες χρειάζονται για να γίνει το cw 1023, το πακέτο πετιέται και η τιμή του cw γίνεται reset. Δηλαδή το cw ορίζεται πάλι στο διάστημα μεταξύ 15 και 31 time slots, άσχετα με το γεγονός ότι όλες οι προηγούμενες προσπάθειες για αποστολή απέβησαν άκαρπες.

ΚΕΦΑΛΑΙΟ 4

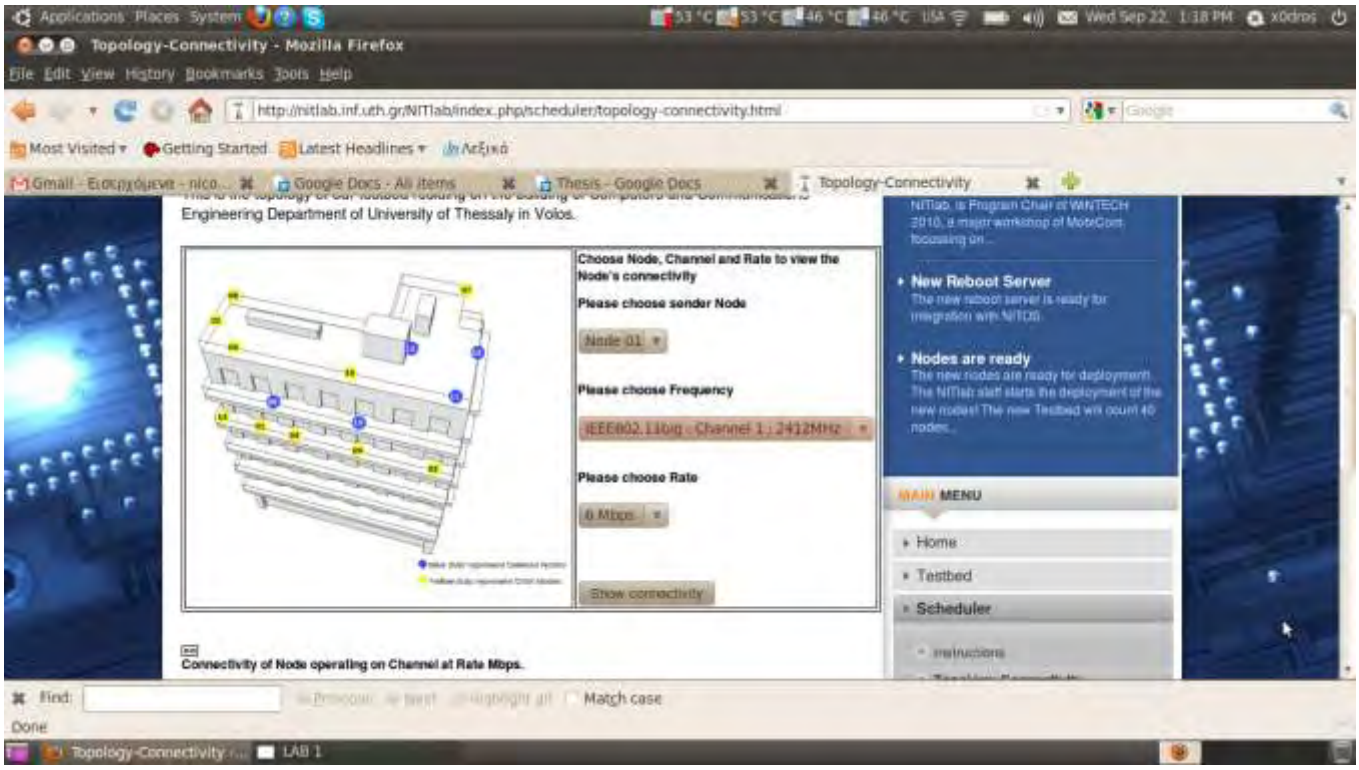
Σύντομη παρουσίαση του NITOS Testbed



Εικόνα 4.1

Το Nitos (Network Implementation Testbed using Open Source code) είναι μια διάταξη ασύρματων κόμβων η οποία μπορεί να χρησιμοποιηθεί για πειράματα και μετρήσεις σε πραγματικές συνθήκες και προσφέρει την ευκαιρία στους ερευνητές να εξετάσουν τα αποτελέσματα της δουλειάς τους, χωρίς την

“προστασία” που προσφέρουν τα προγράμματα προσομοίωσης, εξετάζοντάς τα ως προς τη λειτουργικότητα και απόδοση απέναντι σε προβλήματα που προκύπτουν σε ρεαλιστικά σενάρια και πραγματικές καταστάσεις λειτουργίας. Αποτελείται από 15 κόμβους, οι οποίοι αναμένεται να αυξηθούν στους 40, εγκαταστημένοι στην ταράτσα και στους 2 τελευταίους ορόφους του κτιρίου του τμήματος Μηχανικών Η/Υ Τηλεπικοινωνιών και Δικτύων στην οδό Γκλαβάνη και αναπτύχθηκε από το Nitlab.
Οι κόμβοι αυτοί είναι 2 ειδών, Orbit και Diskless. Οι πρώτοι διαθέτουν σκληρό δίσκο ενώ οι δεύτεροι όχι. Ελέγχονται και λειτουργούν από έναν κεντρικό server μέσω switches. Το σύστημα είναι εξοπλισμένο με εργαλεία τόσο για παροχή δικτυακών υπηρεσιών όσο για τη διαχείριση του Testbed. Στην πρώτη κατηγορία ανήκει το NITOS nfs, ένα δικτυακό λειτουργικό σύστημα βασισμένο σε linux, PXE, εργαλείο που χρησιμεύει για το network booting, DHCP, DNS και APACHE. Στην δεύτερη ανήκουν το OMF (cOntrol and Management Framework), με το οποίο κανείς μπορεί να σχεδιάσει και να ελέγχει τα πειράματά του. Τέλος, το σύστημα παρέχει έναν scheduler ο οποίος αποτελεί ένα εργαλείο ενημέρωσης σχετικά με τη διαθεσιμότητα των κόμβων και των ελεύθερων συχνοτήτων ενώ παρέχει την ταυτόχρονη αξιοποίηση τους προσφέροντας διαμοιρασμό του φάσματος των συχνοτήτων. Μέσω μιας εύχρηστης διαδικτυακής διεπαφής ο ερευνητής μπορεί να δεσμεύσει κόμβους στις επιθυμητές συχνότητες και για την επιθυμητή διάρκεια η οποία για λόγους δικαιοσύνης δεν μπορεί να υπερβεί τις 4 ώρες. Όλα αυτά είναι στη διάθεση του καθένα, αρκεί να μπει στην ιστοσελίδα του Nitlab, http://nitlab.inf.uth.gr, και κάνει εγγραφή και καταχώρηση των στοιχείων που θα του ζητηθούν. Στην παρακάτω εικόνα φαίνεται ένα κομμάτι από την διεπαφή δέσμευσης των κόμβων.



Εικόνα 4.2: Έλεγχος της συνδεσιμότητας των κόμβων, βάσει αριθμού του κόμβου, των συχνοτήτων και τα προσφερόμενα rate.

ΚΕΦΑΛΑΙΟ 5

Σύντομη περιγραφή του Madwifi και τμημάτων του κώδικα που ενδιέφεραν την παρούσα εργασία

Το MADWIFI, (Multiband Atheros Driver for Wifi) είναι ένας driver ασυρμάτων καρτών δικτύου ανοιχτού κώδικα για linux συστήματα. Είναι σχεδιασμένο από μια ομάδα εθελοντών developers για wifi κάρτες με chipset της Atheros και ειδικά τις σειρές ath5xxx και κάτω. Ξεχωρίζει για την ευελιξία, αντοχή και ανοχή σε τροποποιήσεις και , μέχρι πρόσφατα, ενεργή ανάπτυξη και εμπλουτισμό του. Δυστυχώς το τελευταίο διάστημα η ανάπτυξή του έχει σταματήσει και το ενδιαφέρον όσων ασχολούνταν με αυτό έχει στραφεί σε νέους driver ανοιχτού κώδικα όπως ath5k και ath9k. Το τελευταίο είναι η νεότερη έκδοση, και παρέχει υποστήριξη για τις κάρτες με chipset σειράς ath9xxx που λειτουργούν σύμφωνα με το πρωτόκολλο IEEE80211.n.

Ο driver είναι μεν ανοιχτός, αλλά εξαρτάται από το HAL (Hardware Abstraction Layer,) το οποίο είναι ιδιοκτησία της Atheros πράγμα που το εξαιρεί από το να χαρακτηριστεί ανοιχτό λογισμικό, και αποτελεί το ενδιάμεσο επίπεδο μεταξύ του κώδικα του driver και το hardware και μέσω αυτού δρομολογείται η πρόσβαση και επικοινωνία από και προς το hardware.

Ο λόγος που η Atheros δεν επιτρέπει την πρόσβαση των developers στο HAL είναι ότι αντίθετα με πολλές ασύρματες κάρτες, όσες “φορούν” chipset Atheros μπορούν να χρησιμοποιήσουν ένα μεγάλο εύρος συχνοτήτων, πράγμα που μπορεί να έρθει σε σύγκρουση με τους κανονισμούς των ρυθμιστικών αρχών

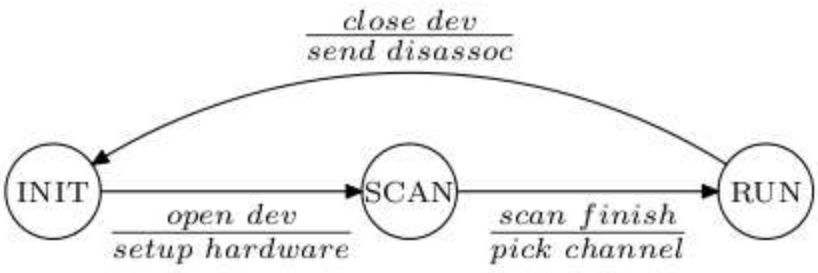
διάφορων χωρών. Αυτό βέβαια μειώνει κάπως την δυνατότητα μεγαλύτερων παρεμβάσεων και πειραματισμών σ' όλο το εύρος των δυνατοτήτων αυτών των συσκευών αλλά ακόμα και στην παρούσα μορφή το Madwifi αποτελεί ένα πολύ χρήσιμο εργαλείο στην έρευνα στον χώρο των ασύρματων δικτύων ως μια ισχυρή και αποδεκτή εναλλακτική στα προγράμματα προσομοίωσης όπως Network Simulator ή ακόμα και Matlab.

Στην παρούσα εργασία, χρησιμοποιήθηκαν συγκεκριμένες λειτουργίες και κομμάτια του κώδικα του Madwifi τα οποία θα επεξηγηθούν στο εδάφιο αυτό.

Association στο Madwifi:

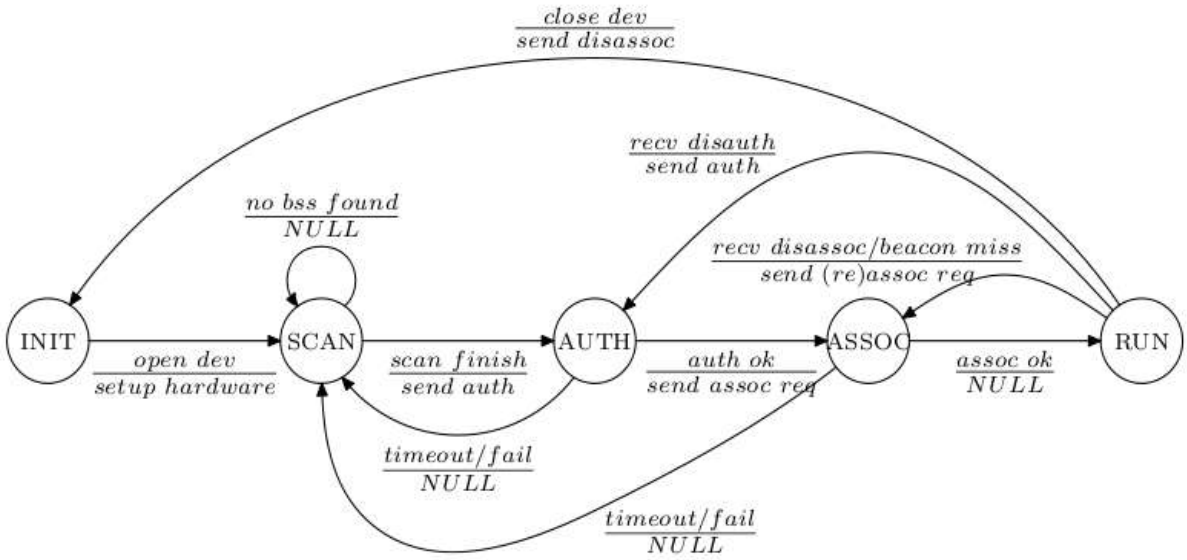
Το association λειτουργεί σύμφωνα με το πρότυπο που περιγράφτηκε στο 2ο κεφάλαιο. Εδώ θα φανεί το πως υλοποιεί το Madwifi αυτήν τη διαδικασία.

Το association χρειάζεται δυο μέλη. Τον σταθμό και το Access Point στον οποίον θα συνδεθεί. Στο AP, από τη στιγμή που θα ξεκινήσει η λειτουργία της κάρτας ο driver αρχικοποιεί την φυσική συσκευή (INIT). Αυτό επιτυγχάνεται με την κλήση της συνάρτησης ath_attach() στο αρχείο ath/if_ath.c. Στη συνέχεια, η ieee80211_create_vap() δημιουργεί ένα εικονικό (virtual) interface δικτύου το οποίο θα λειτουργεί στο mode που έχει καθοριστεί. Όταν "σηκωθεί" το interface π. χ με ifconfig athx up, η κάρτα βγαίνει από την κατάσταση αρχικοποίησης και ξεκινά το scanning. Μετά το τέλος του scan, επιλέγεται το πιο "ήσυχο" κανάλι και εκεί ξεκινά η λειτουργία του AP. Εδώ το νεοσύστατο AP θα ξεκινήσει τη μετάδοση των beacons για να ενημερώσει το περιβάλλον του για την παρουσία του. Όταν, σταματήσει την λειτουργία του, ενημερώνει τους συνδεδεμένους σ' αυτόν σταθμούς στέλνοντας ένα disassociation management πακέτο. Όλα αυτά φαίνονται στο παρακάτω διάγραμμα καταστάσεων.



Εικόνα 5.1

Στην πλευρά του σταθμού, αφού αρχικοποιηθεί η συσκευή, όπως και στο AP, ξεκινά και εδώ η διαδικασία του scanning. Μετά το πέρας του scanning, ο σταθμός επιλέγει με την κλήση της συνάρτησης sta_pick_bss() το δίκτυο εκείνο με το επιθυμητό essid και το μεγαλύτερο rssi. Στη συνέχεια, μπαίνει στην κατάσταση AUTH, όπου ανταλλάσσονται μηνύματα αυθεντικοποίησης. Αν η διαδικασία της αυθεντικοποίησης αποτύχει ή ο σταθμός δεν δεχτεί καμία απάντηση στα επόμενα 5 δευτερόλεπτα καλείται η ieee80211_tx_timeout() και ο σταθμός επιστρέφει στην κατάσταση scanning. Στην περίπτωση επιτυχίας της αυθεντικοποίησης, ο σταθμός μπαίνει στην κατάσταση ASSOC η οποία ξεκινά με την αποστολή ενός ASSOCIATION_REQ μηνύματος στον AP. Αν δεχτεί μήνυμα που να δηλώνει επιτυχημένο association, μπαίνει στην κατάσταση RUN. Αλλιώς, όπως πριν αν λάβει μήνυμα αποτυχίας ή δεν λάβει απάντηση σε 5 δευτερόλεπτα επιστρέφει στην κατάσταση SCAN.
Πλέον ο σταθμός είναι associated με το AP, και μπορεί να στέλνει και να λαμβάνει πακέτα δεδομένων από και προς το δίκτυό του. Παράλληλα, δέχεται management πακέτα που θα στέλνει ανα διαστήματα ο AP του. Κάποια από αυτά μπορεί να είναι disassociation ή disauthentication μηνύματα, οπότε ο σταθμός θα μπει στην κατάσταση ASSOC ή AUTH αντίστοιχα. Όσο ο σταθμός είναι associated, δέχεται ανά 100ms beacons από το AP του. Σε περίπτωση που χάσει 10 συνεχόμενες μεταδόσεις beacons θεωρεί ότι επικοινωνία του με το Access Point έχει σπάσει και στέλνει ένα μήνυμα επανασύνδεσης (Reassociation). Όλα αυτά φαίνονται στο παρακάτω διάγραμμα.



Εικόνα 5.2

Όταν το AP δεχτεί ένα μήνυμα association από κάποιον σταθμό, καλεί την ieee80211_node_join() μέσα από το ieee80211_recv_mgmt() όπου γίνονται κάποιοι αρχικοί έλεγχοι για το αν μπορεί το AP να τον δεχτεί , ώστε να δεσμευτούν πόροι για το εν λόγω σταθμό. Εκεί αφού καταγραφούν τα χαρακτηριστικά του, όπως υποστηριζόμενα rates, δεσμεύεται χώρος γι‘ αυτόν στον πίνακα των κόμβων (node_table) του AP και του αποστέλλεται ένα μήνυμα ASSOCIATION_RESP στο πεδίο status του οποίου καταγράφεται η επιτυχία της αίτησης.

Beacons στο Madwifi

Η δομή των Beacons καθώς και των Probe Responses στο Madwifi όπως ορίζεται στη συνάρτηση ieee80211_beacon_alloc() στο ieee80211_beacon.c είναι η ακόλουθη:

* [8] time stamp
* [2] Beacon interval
* [2] Capability information
* [tlv] ssid
* [tlv] supported rates
* [7] FH/DS parameter set
* [tlv] IBSS/TIM parameter set
* [tlv] country code
* [3] power constraint
* [5] channel switch announcement
* [3] extended rate phy (ERP)
* [tlv] extended supported rates
* [tlv] WME parameters
* [tlv] WPA/RSN parameters
* [tlv] Atheros Advanced Capabilities
* [tlv] AtherosXR parameters
* XXX Vendor-specific OIDs (e.g. Atheros)

Οι αριθμοί δίπλα σε κάθε πεδίο δηλώνουν το μέγεθος του αντίστοιχου πεδίου σε bytes. Σε μερικά από αυτά, τα μεγέθη γράφονται ως tlv, που σημαίνει type length value, δηλαδή η τιμή του μεγέθους του τύπου, αντί για αριθμούς. Το tlv αναφέρεται στο μέγεθος ενός πεδίου που είναι σύνθετο , συνήθως ένα struct, και μιας και τα πεδία των struct μπορεί να αλλαχθούν, το μέγεθός τους δεν είναι αυστηρά καθορισμένο στο παραπάνω πρότυπο, αλλά τροποποιείται ανάλογα με τις αλλαγές που μπορεί να εισάγει ο developer.

Κάθε φορά που έρχεται η στιγμή της μετάδοσης ενός beacon, το HAL στέλνει ένα interrupt και καλείται η συνάρτηση ath_beacon_send (). Από εκεί καλείται η ath_beacon_generate() η οποία δεσμεύει έναν sk_buf για το beacon

και στη συνέχεια καλεί την ieee80211_beacon_update() η οποία ενημερώνει τα δυναμικά πεδία που μπορεί να χρειαστούν μεταβολές. Επιστρέφοντας στην ath_beacon_generate() ο buffer που κρατά το beacon γίνεται map και συγχρονίζεται με τη μνήμη της κάρτας. Όταν γίνει επιστροφή στη ath_beacon_send() παραδίδεται ο buffer του beacon στο HAL και από εκεί στην hardware ουρά για μετάδοση.

BACKOFF και Madwifi

Όπως σε όλες τις ασύρματες κάρτες, έτσι και στις Atheros το Carrier Sensing και backoff λαμβάνουν χώρα αποκλειστικά μέσα στο hardware. Ο λόγος που γίνεται αυτό είναι τα πολύ μικρά χρονικά περιθώρια που επιτρέπουν οι προδιαγραφές της διαδικασίας αυτής όπως περιγράφτηκαν στο κεφάλαιο 2. Η υπεροχή του hardware συγκριτικά με το λογισμικό στην ταχύτητα, εμποδίζει την μετάδοση οποιασδήποτε πληροφορίας σχετικά με το τρέχον μέγεθος του cw να φτάσει στο επίπεδο του driver. Τα μόνα που μπορεί να καθοριστούν μέσα από το driver είναι τα **αρχικά** όριά του, CWmin CWmax, καθώς και αν θα είναι ενεργοποιημένο ή όχι. Αυτά μπορούν να γίνουν στη συνάρτηση ath_txq_setup() όπου ορίζονται τα χαρακτηριστικά των ουρών μετάδοσης. Η διαδικασία είναι by default ενεργοποιημένη ενώ η απενεργοποίησή της γίνεται με την εντολή

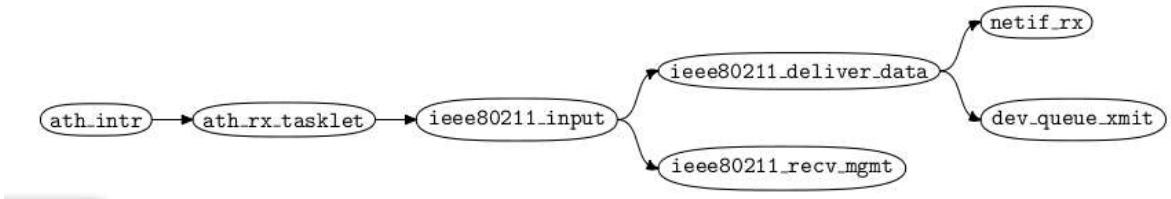
```
qi.tqi_qflags |= HAL_TXQ_BACKOFF_DISABLE;
```

στην συνάρτηση αυτή.

Λήψη πακέτων στο Madwifi

Η λήψη των πακέτων ξεκινά με την κεραία της κάρτας να ανιχνεύει μετάδοση στο κανάλι, όταν η μετάδοση αυτή φτάσει στο hardware αυτό στέλνει ένα interrupt στο πιο πάνω επίπεδο ενημερώνοντας τον driver για λήψη

δεδομένων. Η συνάρτηση που χειρίζεται αυτά τα interrupts είναι η ath_intr() η οποία αφού αποφασίσει για το είδος του interrupt, στη συνέχεια καλεί την ath_rx_tasklet() όπου δεσμεύεται ένας sk_buffer για το πακέτο που μόλις ήρθε. Αυτό το buffer στη συνέχεια παραδίδεται στην ieee80211_input() όπου ανάλογα με το είδος του πακέτου, data ή management, παραδίδεται στις ieee80211_deliver_data() ή ieee80211_recv_mgmt() αντίστοιχα. Η πρώτη μεταβιβάζει το πακέτο στον πυρήνα του λειτουργικού και η άλλη προχωρεί στις απαραίτητες ενέργειες που ορίζονται από το είδος του management πακέτου. Στην παρακάτω εικόνα φαίνονται διαγραμματικά τα όσα μόλις ειπώθηκαν.

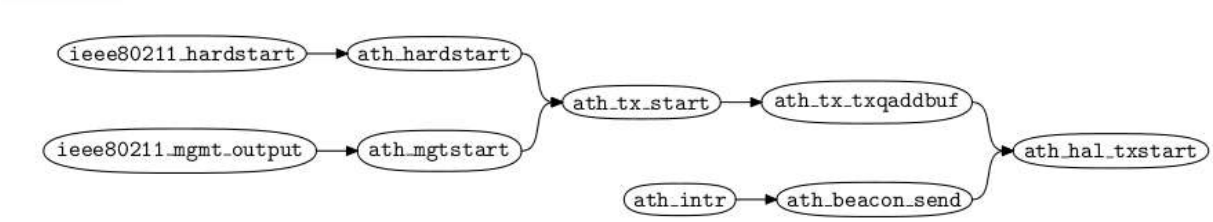


Εικόνα 5.3

Μετάδοση πακέτων Madwifi

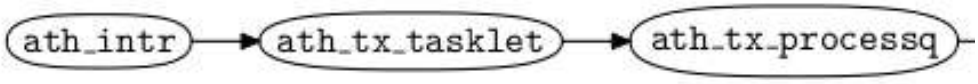
Στην μετάδοση, ο πυρήνας καλεί την ieee80211_hardstart() με όρισμα έναν sk_buffer όπου υπάρχουν τα περιεχόμενα του προς μετάδοση πακέτου. Το πακέτο είναι ένα Ethernet πακέτο αφού ο πυρήνας δεν κάνει διαχωρισμό μεταξύ Ethernet ή wireless πακέτα. Αυτή, με τη σειρά της καλεί την ath_hardstart(). Η τελευταία καλεί την ieee80211_encap() η οποία επιστρέφει τον sk_buffer με τον οποίον καλέστηκε, αφού πρώτα τον έχει τροποποιήσει σε 80211 πακέτο. Πρόσθεσε δηλαδή τον wireless header και αν ήταν απαραίτητο έκανε fragmentation στο πακέτο. Στην τελευταία περίπτωση ο sk_buffer που επιστρέφεται είναι ο πρώτος μιας λίστας από buffers/ πακέτα που κατατμήθηκαν σε fragments. Στη συνέχεια η ath_tx_start() κυπτογραφεί το πακέτο, αν χρειαστεί, και το κάνει map σε έναν DMA (Direct Memory Access) buffer και επιλέγει την ουρά μετάδοσής του πακέτου ανάλογα με την προτεραιότητα αποστολής του, αν υπάρχουν διαφορετικές ουρές λόγω ενεργοποίησης της

υποστήριξης του Quality of Service. Τέλος, η ath_tx_addbuff() εισάγει το πακέτο στην ανάλογη ουρά και ειδοποιεί το HAL να ξεκινήσει τη μετάδοση.
Τα management πακέτα, ξεκινούν τη μετάδοσή τους από την ieee80211_mgmt_output().
Όλα αυτά φαίνονται στο παρακάτω διάγραμμα.



Εικόνα 5.4

Μετά από επιτυχημένη αποστολή πακέτου, το hardware σηκώνει πάλι ένα interrupt το οποίο μέσω HAL θα φτάσει στον driver που θα το χειριστεί με την ath_intr(). Έπειτα, καλείται η ath_tx_tasklet() και στη συνέχεια η ath_tx_processq οι οποίες ανανεώνουν κάποια στατιστικά σχετικά με τις μεταδόσεις των πακέτων. Παρακάτω φαίνεται σε διαγραμματική μορφή η αλυσίδα που περιγράφτηκε.



Εικόνα 5.5

ΚΕΦΑΛΑΙΟ 6

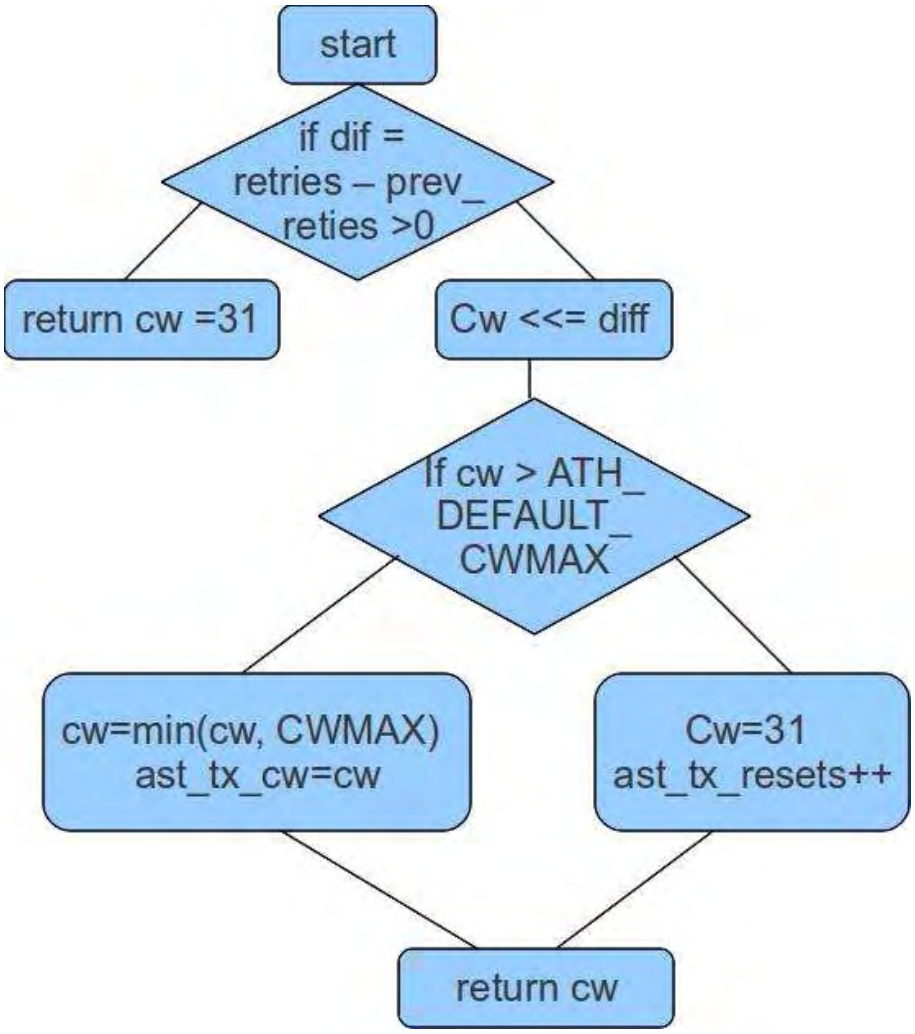
SoftBackoff: Η Δική μου εργασία πάνω στο Backoff

Όπως ειπώθηκε πριν, τα Carrier Sensing και Backoff είναι κρίσιμες λειτουργίες των ασύρματων καρτών των οποίων όμως την αποκλειστική ευθύνη έχει το hardware και όλη η διαδικασία και τα δεδομένα που παράγονται από αυτήν είναι πλήρως αδιαφανή στον driver. Ο driver, ενημερώνεται για την επιτυχία ή την αποτυχία αποστολής ενός πακέτου μόνο μετά από το πέρας της διαδικασίας του Backoff και ποτέ κατά τη διάρκειά του.
Το παραπάνω, δεν μου ήταν γνωστό όταν μου ήρθε η ιδέα για ένα σχήμα association που θα εκμεταλλεύεται τα δεδομένα και στατιστικά που προκύπτουν από το Backoff. Γι' αυτόν τον λόγο, πολύ μεγάλο μέρος του χρόνου μου αναλώθηκε στην προσπάθεια να βρω σε ποιο σημείο του κώδικα του Madwifi φαίνονται αυτά. Προσπάθεια που απέβη φυσικά άκαρπη μιας και μετά από αρκετό καιρό ψάξιμο, σχεδόν στα τυφλά, και έρευνα στον κώδικα του Madwifi ο οποίος είναι αρκετά μεγάλος και πολύ κακώς τεκμηριωμένος, και πολλά emails στη λίστα του, διαπίστωσα ότι ο driver δεν "βλέπει" τα όσα γίνονται στο Backoff. Δεν φαίνεται πουθενά πώς, πότε και γιατί αλλάζει το τρέχον μέγεθος του Contention window. Για την ακρίβεια δεν φαινόταν πουθενά το Contention window, παρά μόνο τα αρχικά όριά του CWmin και CWmax.
Μετά από συζήτηση αυτών των προβλημάτων με τον κ. Κοράκη, εκείνος πρότεινε αντί να βασιζόμαστε σε δεδομένα που ούτως ή άλλως δεν μπορούμε να δούμε,

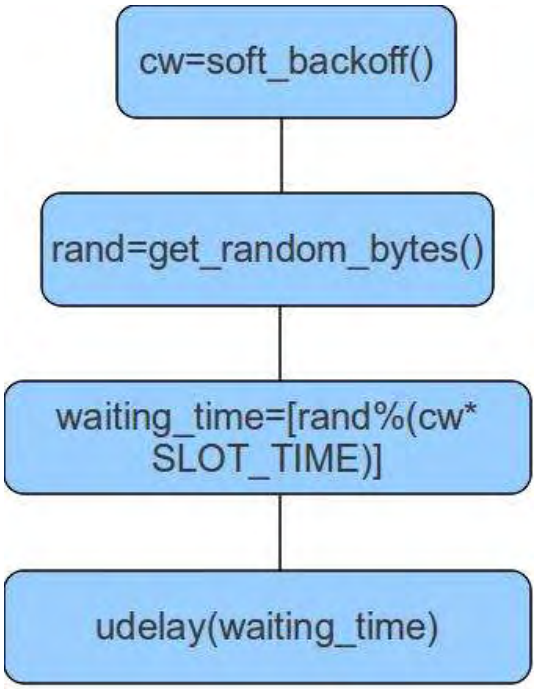
να απενεργοποιήσω το backoff και να τον προσομοιώσω (emulate) σε επίπεδο driver. Να φτιάξω δηλαδή μια software έκδοση του Backoff, το Soft Backoff. Έτσι, η προσπάθειά μου επικεντρώθηκε αρχικά σ' αυτήν την κατεύθυνση.
Για να προσομοιωθεί το Backoff, χρειαζόμαστε πρώτα απ' όλα μια ένδειξη που να μας δηλώνει ή τουλάχιστον να υπονοεί σύγκρουση, ώστε με βάση αυτή να αυξομειώνουμε το μέγεθος του cw. Μία τέτοια ένδειξη είναι μια στατιστική μεταδόσεων που ο κρατάει driver, και ανανεώνεται μετά από κάθε (επιτυχημένη) μετάδοση ή packet drop. Αυτή η μεταβλητή ονομάζεται sc_stats.ast_tx_longretry, και είναι ένα πεδίο του struct ath_stats όπως ορίζεται στο αρχείο if_athioctl.h και ανανεώνεται στη συνάρτηση ath_tx_processq() (Εικόνα 4.5). Η μεταβλητή αυτή μετρά τις επαναποστολές, στις οποίες προχωρά (μάλλον έχει ήδη προχωρήσει) η κάρτα. Μια επαναποστολή υπονοεί πάντα μια αποτυχία προηγούμενης αποστολής για διάφορους λόγους, όπως αδυναμία εύρεσης δέκτη ή σύγκρουση. Συνεπώς, είναι μια μετρική στην οποία μπορούμε να βασίσουμε την αυξομείωση του μήκους του Contention window.
Το δεύτερο θέμα που πρέπει να λυθεί, είναι σε ποιο σημείο της αλυσίδας μετάδοσης πακέτων πρέπει να γίνει το backoff. Για να είμαστε όσο πιο κοντά γίνεται στο κανονικό hardware backoff, θα πρέπει το σημείο αυτό να βρίσκεται όσο το δυνατόν πιο κοντά στο τέλος της αλυσίδας, στο σημείο δηλαδή που παραδίδεται το πακέτο στο hardware. Στην εικόνα 4.4 βλέπουμε ότι αυτό το σημείο είναι η συνάρτηση ath_hal_tx_start(), όμως όπως είπαμε ο developer δεν μπορεί να "πειράξει" το HAL, και η εν λόγω συνάρτηση ανήκει στο χώρο του τελευταίου. Συνεπώς, τοποθετούμε το Soft Backoff ένα στάδιο πιο πριν, δηλαδή στο τέλος της ath_tx_addbuf() πριν την κλήση της ath_hal_tx_start().
Ένα άλλο ερώτημα που προέκυψε ήταν αυτό. Με ποίον τρόπο θα "περιμένουμε"; Πως θα σταματήσει για κάποιο ορισμένο χρόνο η παράδοση των πακέτων στο HAL; Στην αρχή δοκιμάστηκε η λύση των timers. Υπήρχε ωστόσο ένα πρόβλημα σ' αυτό: οι timers παίρνουν όρισμα jiffies που είναι της τάξης των mili second, ενώ το backoff λειτουργεί με μεγέθη της τάξης των micro second. Γι' αυτό έπρεπε να βρεθεί άλλος τρόπος για να επιτευχθεί η αναμονή. Μετά από αρκετό ψάξιμο και ενασχόληση με κάποια tricks που ίσως μπορούσαν να "ξεγελάσουν" τον timer και άλλες λύσεις που όμως θα εξαρτιόνταν από τις ιδιότητες της εκάστοτε μηχανής, όπως ταχύτητα του επεξεργαστή για παράδειγμα, βρέθηκε ως λύση, η

συνάρτηση udelay() του πυρήνα του linux. Η συνάρτηση αυτή παγώνει την εκτέλεση ενός κώδικα για διαστήματα τάξης micro second, και ως εκ τούτου ταιριάζει απόλυτα στις ανάγκες αναμονής για την προσομοίωση του Backoff.

Η υλοποίηση του Softackoff είναι πολύ απλή. Στη συνάρτηση ath_tx_addbuf() πρίν την κλήση της ath_hal_tx_start(), καλείται η soft_backoff(). Εκεί, ελέγχεται αν ο αριθμός των επαναποστολών (sc_stats.ast_tx_longretry) είναι ίδιος ή μεγαλύτερος από την προηγούμενη φορά που καλέστηκε η συνάρτηση (Ο αριθμός των προηγούμενων retries αποθηκεύεται κάθε φορά σε ένα νέο πεδίο του struct ath_stats, το ast_tx_prevretry). Αν είναι ίδιος, επιστρέφεται cw = 31 αλλιώς υπολογίζεται το cw = $2^{(longretry - prevretry)} - 1$. Αν το cw βγει μεγαλύτερο από την τιμή ATH_DEFAULT_CWMAX η οποία ισούται με 1023, το μέγεθος του παραθύρου γίνεται reset στο 31, αυξάνεται η τιμή του ast_tx_resets, που κρατά την τιμή των resets, κατά 1 και η νέα τιμή του cw αποθηκεύεται στο πεδίο ast_tx_cw.
Η soft_backoff() επιστρέφει το τρέχον μήκος του παραθύρου, το οποίο στην ath_tx_addbuf(), αφού γίνει mod με έναν τυχαίο αριθμό ο οποίος παράγεται μέσω της συνάρτησης του πυρήνα get_random_bytes(), ώστε ο νέος αυτός αριθμός να έχει μέγεθος μέχρι cw, πολλαπλασιάζεται με το time slot το οποίο στα δίκτυα 80211.g ισούται με 9μs. Στη συνέχεια, χρησιμοποιώντας την udelay(), εισάγουμε μια καθυστέρηση ίση με rand_number $\in$[cwmin, cw] x TIME_SLOT micro seconds. Με αυτόν τον τρόπο καθυστερούμε - τεχνητά - την μετάδοση του επόμενου πακέτου από τον driver. Η λογική που περιγράψαμε φαίνεται στα παρακάτω διάγραμμα:



Εικόνα 6.1: soft_backoff()



Εικόνα 6.2: ath_tx_addbuf, πριν την κλήση της ath_hal_tx_start()

Το παραπάνω σχήμα, έχει κάποια εμφανή προβλήματα:

- Το πρώτο πρόβλημα είναι ο χρόνος! Ως γνωστόν, το hardware είναι πού πιο γρήγορο από το λογισμικό, και τα δεδομένα που χρειάζεται το soft_backoff αργούν να φτάσουν από την κάρτα στον driver καθιστώντας τη λειτουργία του emulation πολύ πιο αργή συγκριτικά με το πραγματικό backoff.
- Το δεύτερο έχει να κάνει με τα δεδομένα που τελικά φτάνουν στο επίπεδο του driver. Η μεταβλητή που κρατά τις επαναποστολές (sc_stats.ast_tx_longretry), δεν ανανεώνεται σε κάθε αποτυχία αποστολής αλλά μετά από μια οριστική κατάσταση, δηλαδή επιτυχημένη αποστολή ή απόρριψη πακέτου. Αυτό έχει ως αποτέλεσμα, η συνάρτηση soft_backoff() να υπολογίζει το μήκος του cw όπως αυτό (θα) ήταν κάποιες χρονικές στιγμές πριν και όχι τώρα. Η καθυστέρηση που εισάγει όμως, αφορά την παρούσα χρονική στιγμή! Με άλλα λόγια, δανειζόμαστε χρόνο από την επόμενη αποστολή! Το πακέτο στέλνεται από την κάρτα ούτως η άλλως

και αν προκύψει αποτυχία κάνουμε backoff στην επόμενη απόπειρα αποστολής, ή μάλλον καλύτερα, την επόμενη φορά που θα παραδώσει ο driver το πακέτο στο hardware!

- Ένα άλλο μεγάλο ζήτημα στο οποίο υστερεί το soft_backoff σε σχέση με το πραγματικό, είναι η παραλληλοποίηση που προσφέρει η ύπαρξη 2 ξεχωριστών οντοτήτων, το hardware από τη μια και ο driver από την άλλη, η κάθε μια από τις οποίες κάνει ανεξάρτητα τη δουλειά της χωρίς να περιμένει η μια την άλλη. Στο backoff, το πακέτο παραδίδεται στο HAL και από εκεί στη συσκευή ενώ η τελευταία είναι υπεύθυνη για την τελική αποστολή του. Σε περίπτωση που δεν είναι δυνατή η μετάδοση, το hardware περιμένει στα πλαίσια του backoff μέχρι να καταφέρει να στείλει το πακέτο, χωρίς αυτό να επηρεάζει τη λειτουργία του driver ο οποίος εν τω μεταξύ μπορεί να δεχτεί κι άλλα πακέτα από τον πυρήνα και να τα παραδώσει στην κάρτα χωρίς καμία ανάγκη αναμονής. Στο Soft Backoff όμως λόγω αδυναμίας πρόσβασης στο hardware, η αναμονή αυτή μεταφέρθηκε στον driver προσθέτοντας μια καθυστέρηση στη ροή παράδοσης των πακέτων στη συσκευή.
- Τέλος, Η διαδικασία του Backoff όπως περιγράφτηκε πριν, είναι άρρηκτα συνδεδεμένη με το Carrier Sensing. Στο δικό μας μοντέλο, επειδή το CS πραγματοποιείται στο hardware και τα σχετικά μ’ αυτό δεδομένα δεν φτάνουν στο driver, η αντίστροφη μέτρηση μέχρι την αποστολή του πακέτου δε σταματά όταν στο κανάλι υπάρχει άλλη μετάδοση.
  Η επίδραση αυτής της παρατήρησης θα φανεί καλύτερα στα πειραματικά αποτελέσματα .

Αυτά, συν την απενεργοποίηση του φυσικού backoff πράγμα που θα οδηγήσει σε παραπάνω επαναποστολές, αναμένεται να επιβαρύνει το σύστημα και να μειώσει την ταχύτητα και την απόδοση του.

ΚΕΦΑΛΑΙΟ 7

Το Νέο σχήμα Association

Όπως ειπώθηκε στο κεφάλαιο 2, η μετρική βάσει τις οποίας ένας σταθμός επιλέγει το AP στο οποίο θα συνδεθεί είναι το rssi. Η μετρική αυτή έχει κάποια ελαττώματα, τα οποία εξηγήθηκαν και πριν, τα οποία παρακίνησαν την προσπάθεια εύρεσης νέων κριτηρίων association. Σε μια από αυτές τις προσπάθειες, στην εργασία αυτή, προτείνεται η αντικατάστασή της από μια άλλη μετρική η οποία θα λαμβάνει υπόψιν της τόσο το ανοδικό και καθοδικό κανάλι, όσο και τις συνθήκες θορύβου και ποιότητας καναλιού γύρω από τους σταθμούς και το AP.
Η μετρική αυτή, χρησιμοποιεί δεδομένα που παράγονται (ή μάλλον συνάγονται) από τη διαδικασία του Backoff όπως εξηγήθηκε στο προηγούμενο κεφάλαιο, δηλαδή το μήκος του Contention Window και ο αριθμός των resets του μήκους αυτού κατά τη διάρκεια της λειτουργίας του AP και των associated σ' αυτό σταθμών, καθώς και το rate με την οποία δέχονται ή μεταδίδουν δεδομένα οι σταθμοί και το AP. Αυτή η μετρική προκύπτει και ανανεώνεται σε γύρους κάθε 3.5 περίπου δευτερόλεπτα, όσο χρόνο δηλαδή θα κάνει η συσκευή να στείλει 35 beacons.

Ακολουθεί ο αλγόριθμος υπολογισμού της μετρικής αυτής καθώς και της συλλογής των δεδομένων από τα οποία προκύπτει.

1. Ανά κάθε 35 beacons που στέλνει το AP, ζητά από τους σταθμούς που είναι associated σ' αυτό να στείλουν τα rates με την οποία μετέδωσαν την τελευταία φορά, τον γύρω στον οποίον βρίσκονται, τα μήκη των cw και τα resets τους.

2. Οι σταθμοί αυξάνουν τον μετρητή γύρων τους και στέλνουν πίσω στο AP, τα δεδομένα που τους ζητήθηκαν και συνεχίζουν τη ζωή τους.

3. Τα δεδομένα αυτά αποθηκεύονται στο AP, σ' έναν πίνακα κόμβων.

4. Πριν την έναρξη κάθε γύρου, σαρώνεται ο παραπάνω πίνακας και υπολογίζεται η μετρική ως εξής:
   - my_metricDownLink = [ap_CW + ap_Resets] / ap_rate
   - my_metricUpLink =
   $$\sum_{i}^{N} \frac{(rounds_{ap} - rounds_{stai}) * (cw_{stai} + restes_{stai})}{rate_{stai}}$$
   - my_metric = my_metricDownlink + my_metricUplink.

5. Κάθε φορά που το AP δέχεται ένα Probe Request, μεταδίδει την μετρική αυτή στο Probe Response του. Έτσι, ο σταθμός που έστειλε την αίτηση μαθαίνει για το μέγεθος αυτής της μετρικής.

Υλοποίηση του νέου σχήματος:

1.
Οι σταθμοί που συμμετέχουν στη συλλογή των απαραίτητων δεδομένων, ειδοποιούνται κάθε 35 Beacons. Η ειδοποίηση αυτή γίνεται με τον εξής τρόπο. Το AP, στη συνάρτηση ieee80211_beacon_update(), όπου ενημερώνονται τα δυναμικά μέρη του beacon, αλλάζει το πεδίου wh->i_dur του wireless header από μηδέν ,που είναι η συνηθισμένη τιμή, σε ένα.

2.
Έτσι οι σταθμοί όταν λάβουν το συγκεκριμένο beacon και εξετάσουν το πεδίο αυτό, θα καταλάβουν ότι ένας νέος γύρος ανταλλαγής δεδομένων έχει έρθει.

Όταν λάβουν αυτό το Beacon, στη συνάρτηση ieee80211_recv_mgmt(), ελέγχουν αν έχει σταλεί από το AP με το οποίο είναι associated και αν το πεδίο wh->i_dur του wireless header είναι ίσο με 1, εφόσον ισχύουν και τα 2 τότε με την εντολή IEEE80211_SEND_MGMT(ni, IEEE80211_FC0_SUBTYPE_ROUND_RESP, 0) στέλνουν την απόκρισή τους ώς ένα management πακέτο τύπου ROUND_RESP στην (έμμεση) αίτηση έναρξης του γύρου. Με την εντολή αυτή ο έλεγχος μεταφέρεται στην ieee80211_send_mgmt() όπου αυξάνεται ο αριθμός των γύρων στο sc->sc_stats.ast_tx_myrounds, πεδίο που προστέθηκε στ struct ath_tx_stats και κρατά τους γύρους στους οποίους συμμετέχει ο σταθμός, εγγράφονται στα κατάλληλα πεδία οι γύροι αυτοί, το cw, τα resets, και το rate του σταθμού. Έπειτα το πακέτο αυτό παραδίδεται στο HAL για να μεταδοθεί. Το rate που στέλνεται είναι αυτό με το οποίο ο σταθμός έστειλε τελευταία φορά κάποιο πακέτο και αποθηκεύεται στην μεταβλητή ic->ic_rates, στο οποίο ανατίθεται η τιμή της μεταβλητής ni->ni_rates.rs_rates[ni->ni_txrate] στη συνάρτηση ieee80211_encap().
Να σημειωθεί εδώ ότι ο αρχικός σχεδιασμός του αλγορίθμου δεν περιλάμβανε την τροποποίηση του πεδίου i_dur , και την μετάδοση των beacons ως σημείο αναφοράς, αλλά προέβλεπε ένα σχήμα συμμετρικό, όπου μετά από το expire κάποιου timer που είχε σχεδιαστεί γι’ αυτόν το σκοπό, το AP θα έστελνε ένα management πακέτο τύπου IEEE80211_FC0_SUBTYPE_ROUND_REQ, με το οποίο θα σηματοδοτούσε την έναρξη του νέου γύρου. Ο timer αυτός, έπρεπε να βρίσκεται στο αρχείο if_ath.c και να ξεκινά την έναρξή του στη συνάρτηση ath_attach() η οποία είναι η συνάρτηση που αρχικοποιεί την ασύρματη συσκευή, και να διαγράφεται στην ath_detach() η οποία είναι συμμετρική της ath_attach() όταν πρόκειται να σταματήσει η λειτουργία της κάρτας. Όμως η οποιαδήποτε κλίση συνάρτησης στα πλαίσια του timer, είτε για απευθείας μετάδοση του ROUND_REQ είτε για αλλαγή τιμής κάποιας μεταβλητής που θα σηματοδοτούσε την αποστολή της αίτησης οδηγούσε στο πάγωμα του driver. Γι ‘αυτό και αποφάσισα να καταφύγω στη λύση της τροποποίησης του wireless header. Επίσης, το IEEE80211_FC0_SUBTYPE_ROUND_REQ, είναι μια δική μου προσθήκη στο σύνολο των management πακέτων όπως ορίζονται στο αρχείο ieee80211_var.h γραμμή 127. Για την τελική αποστολή ενός management πακέτου είναι απαραίτητη και η τροποποίηση της ieee80211_send_mgmt() στο

αρχείο ieee80211_outout.c, και προσθήκη μιας ακόμα συνθήκης στο switch - case block όπου τροποποιούνται τα πακέτα ανάλογα με το SUBTYPE τους.

3.
Τα δεδομένα που τελικά φτάνουν στο AP, αφού ελεγχθούν ότι προέρχονται από τους σταθμούς του δικτύου που αυτό ορίζει, αποθηκεύονται στα αντίστοιχα πεδία ενός πίνακα κόμβων όπου κρατούνται οι συνδεδεμένοι με το AP κόμβοι. Κάθε θέση του πίνακα αυτού κρατά ένα struct με τα παρακάτω πεδία:

```
 struct ieee80211_node *my_ni;
u_int8_t     mynd_macaddr[IEEE80211_ADDR_LEN];
u_int8_t     mynd_bssid[IEEE80211_ADDR_LEN];
u_int32_t    mynd_rate;
u_int32_t    mynd_roundsin;
u_int32_t    mynd_resets;
u_int32_t    mynd_cw;
u_int8_t    not_empty;
```

Τα οποία με τη σειρά είναι, ieee80211_node, το mac address του κόμβου, το mac address του δικτύου, το rate του κόμβου, οι γύροι στους οποίους συμμετέχει ο κάθε κόμβος, τα resets, το cw, και ένας δείκτης για το αν η θέση του πίνακα είναι κατειλημμένος ή όχι. Το τελευταίο έχει μόνο προγραμματιστική αξία και βοηθά στην προσπέλαση του πίνακα.
Όταν ένας σταθμός γίνεται associated μ' ένα AP, πράγμα που γίνεται με την κλήση της συνάρτησης ieee80211_node_join() στην πλευρά του access point, το AP ελέγχει αν ο σταθμός αυτός βρίσκεται στον παραπάνω πίνακα, αν όχι, τον προσθέτει βρίσκοντας την επόμενη άδεια θέση του πίνακα και τον τοποθετεί εκεί. Αλλιώς δεν κάνει τίποτα μιας και η ieee80211_node_join() πιθανώς να καλέστηκε μετά από κάποια αίτηση επανασύνδεσης (Reassociation). Ο πίνακας αυτός έχει συγκεκριμένο μέγεθος (20 θέσεις) και δεν μπορεί να δεχτεί παραπάνω από τον αριθμό αυτό σταθμούς. Αν δεχτεί κάποια επιπλέον αίτηση association, στέλνει πίσω ένα μήνυμα IEEE80211_REASON_ASSOC_TOOMANY, αναγκάζοντας τον σταθμό να αναζητήσει νέο access point.

4.

Στην αρχή κάθε γύρου, ανά 35 beacons δηλαδή, μέσα από την ieee80211_beacon_update() καλείται η συνάρτηση calc_metric() η οπία υπολογίζει την νέα μετρική. Η εν λόγω συνάρτηση καλείται με όρισμα το node που αντιπροσωπεύει το ίδιο AP, υπολογίζει το my_merticDownlink και στη συνέχεια το my_metricUplink προσπελαύνοντας τα στοιχεία το πίνακα των κόμβων μ' ένα for loop. Η τιμή που επιστρέφεται αποθηκεύεται στο πεδίο ic_mymetric του struct ieee80211com και θα μεταδοθεί όταν το AP δεχτεί κάποιο Probe Request.

Η calc_metric(), καλείται να υπολογίσει μια μετρική η οποία έχει μία πράξη διαίρεσης. Μιας και στο kernel space, όπου ανήκουν οι συναρτήσεις του driver, δεν υπάρχουν δεκαδικοί αριθμοί, το αποτέλεσμα της διαίρεσης θα είναι πάντα το ακέραιο μέρος της διαίρεσης αποκόβοντας τα δεκαδικά. Για να πετύχουμε μια ακρίβεια εκατοστών στις διαιρέσεις μας, πριν τη διαίρεση ολισθαίνουμε τον διαιρετέο κατά 10 θέσεις, όσα bits δηλαδή χρειάζονται για να αναπαραστήσουμε 2 δεκαδικά ψηφία, αριστερά και κάνουμε τη διαίρεση. Στον δέκτη ο αριθμός αυτός αναλύεται στο ακέραιο και δεκαδικό μέρος με διαδοχικές ολισθήσεις δεξιά και αριστερά όπως θα δείτε στις συναρτήσεις int_part() και double_part() που βρίσκονται στο παράρτημα.

5.

Όπως περιγράφτηκε στο κεφάλαιο 3, στο beacon υπάρχει ένα πεδίο με το όνομα country code. Το πεδίο αυτό έχει μέγεθος όσο ένα struct ieee80211_ie_country το οποίο έχει την παρακάτω δομή:

```
 u_int8_t country_id;
 u_int8_t country_len;
 u_int8_t country_str[3];
 u_int8_t
country_triplet[IEEE80211_COUNTRY_MAX_TRIPLETS * 3];
```

Στην παραπάνω δομή προσθέσαμε το πεδίο `u_int32_t country_mymetric`, στο οποίο θα αναθέτουμε το metric που έχουμε υπολογίσει. Στη συνάρτηση ieee_80211_send_mgmt(), στο case PROBE_RESP καλείται η συνάρτηση ieee80211_add_country() και εκεί γίνεται η ανάθεση της μετρικής στο country_metric. Έτσι, όταν ο σταθμός λάβει το PROBE_RESP θα μπορέσει να δει το my_metric μέσα στο αντίστοιχο πεδίο του ieee80211_ie_country struct και να λάβει την απόφαση σύνδεσης σεaccess point ανάλογα με το μέγεθός αυτό.

ΚΕΦΑΛΑΙΟ 8

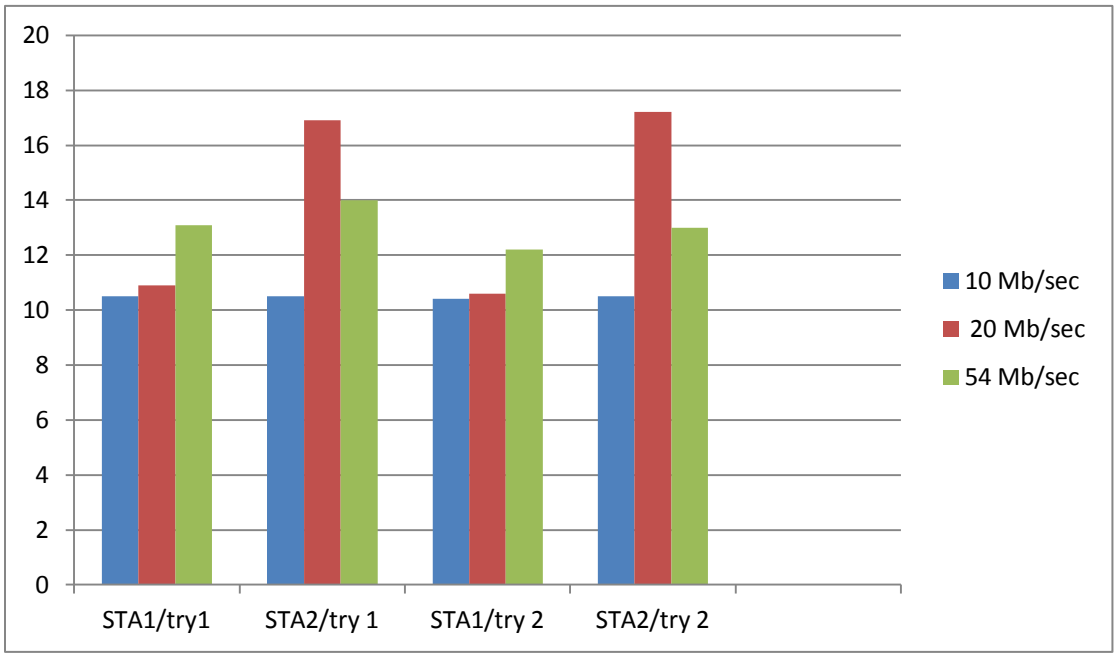
Πειραματικά Αποτελέσματα

3 κύρια σετ (A, B και C) πειραμάτων πραγματοποιήθηκαν. Ακολουθεί η περιγραφή της κάθε μιας καθώς και σχολιασμός στο τέλος κάθε σετ.

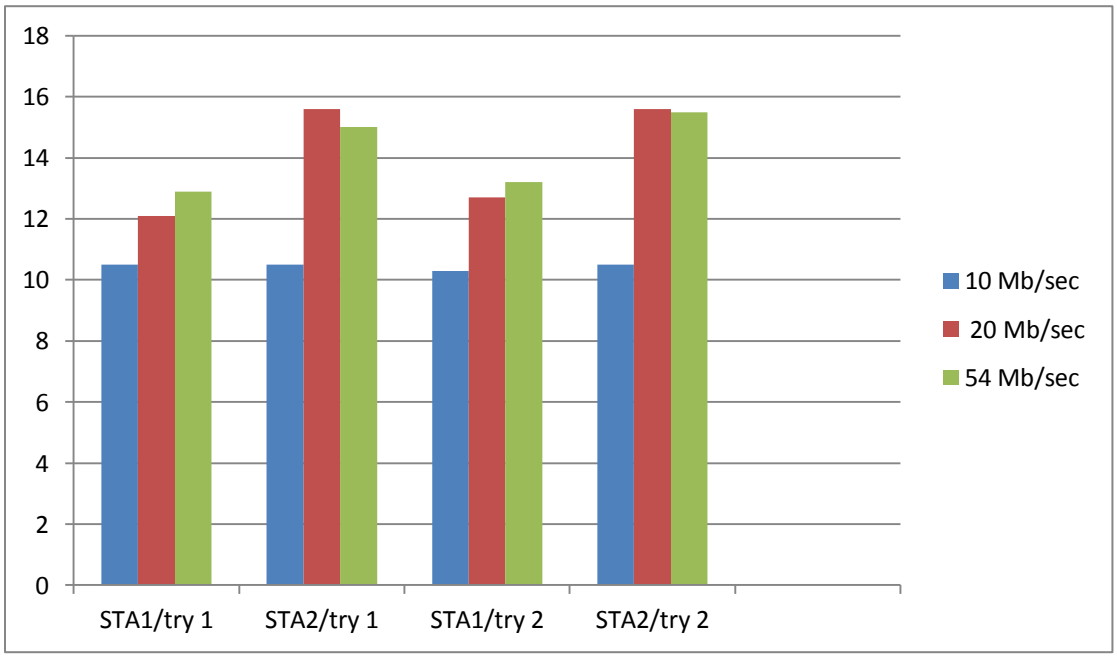
Α) ΔΥΟ ΣΤΑΘΜΟΙ (LAPTOP) ΕΝΑΣ ACCESS POINT (PC)

Το πρώτο, έλαβε χώρα στις 14 Ιουλίου 2010 μεταξύ 2 π. μ. και 5.30 π. μ. με τη χρήση ενός επιτραπέζιου υπολογιστή (AP) εφοδιασμένο με PCI κάρτα TP-LINK με chipset Atheros 5212 και 2 laptops με κάρτες PCMCIA D-LINK και miniPCI, πάλι με chipset Atheros 5212. 6 πειράματα πραγματοποιήθηκαν με τη χρήση του εργαλείου iperf.
Το κάθε πείραμα είχε διάρκεια 120 sec και οι κάρτες λειτουργούσαν στο 80211.g.
Εξετάστηκαν οι περιπτώσεις:
απενεργοποιημένο Backoff, Soft Backoff και Φυσικό Backoff με 2 προσπάθειες (try1 και try2) η κάθε μια.
Αυτά τα 6 πειράματα χωρίζονται σε 2 ομάδες. Στην πρώτη, οι σταθμοί βρίσκονταν στον ίδιο χώρο με το AP ενώ στη δεύτερη, το AP σ' ένα δωμάτιο και οι σταθμοί σ' ένα άλλο. Το κάθε πείραμα εξέτασε το throughput για 3 ξεχωριστά rates: 10, 20 και 54 Mbps.
Στην επόμενη σελίδα φαίνονται τα αποτελέσματα.

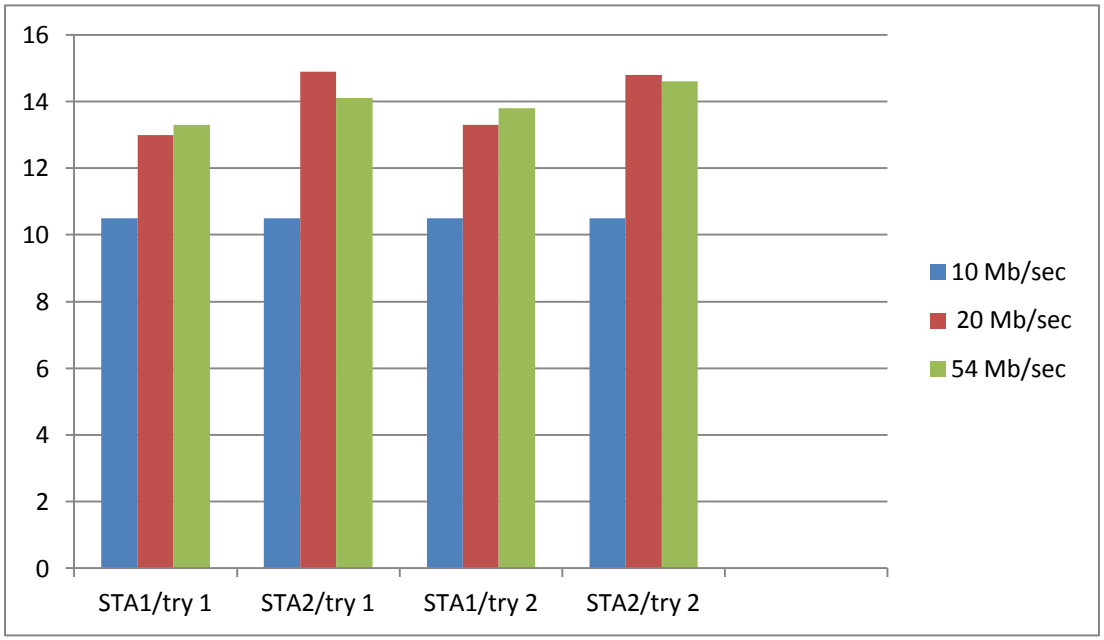
Πείραμα i: Φυσικό Backoff. Σταθμοί και AP σε κοντινοί απόσταση μεταξύ τους.



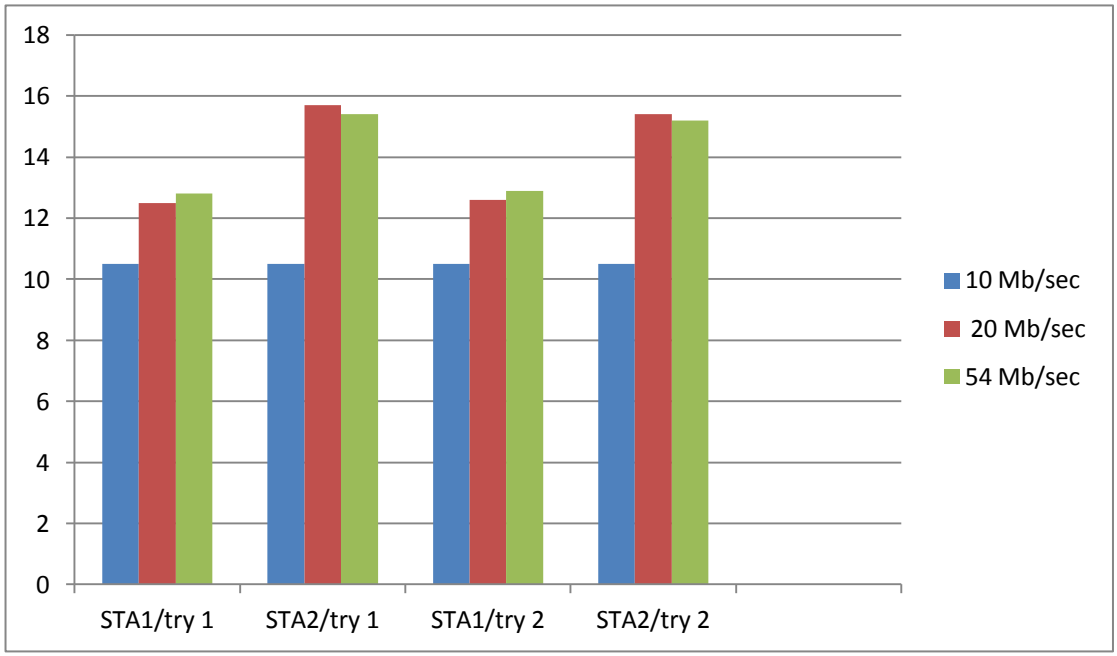
Πείραμα ii: Φυσικό Backoff. Σταθμοί σε κοντινοί απόσταση μεταξύ τους – AP σε άλλο δωμάτιο.



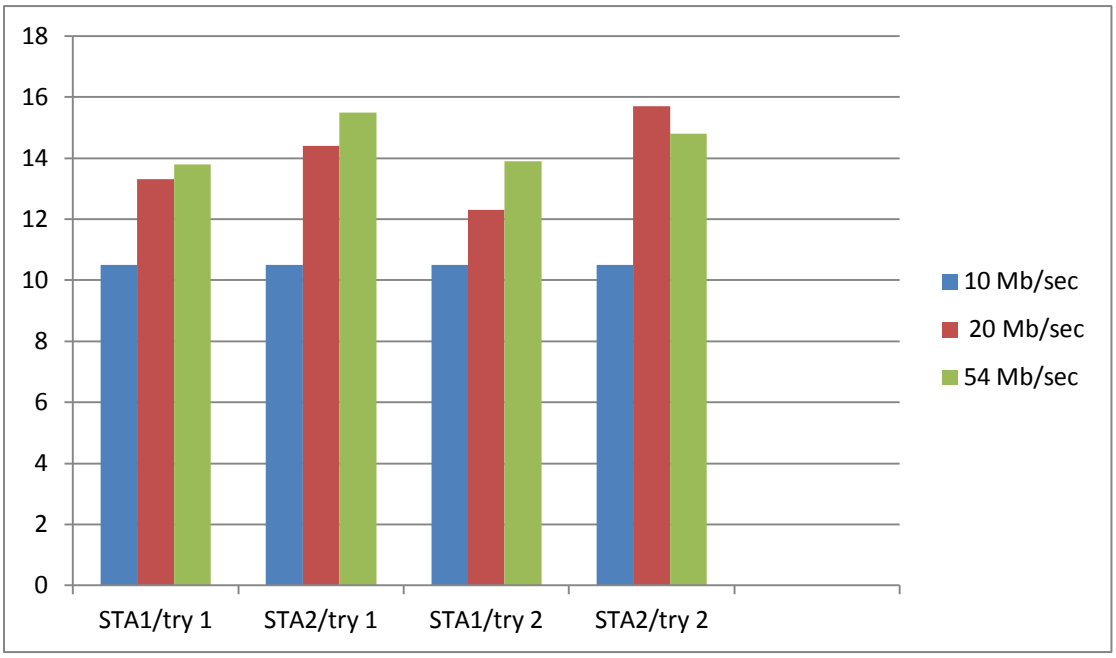
Πείραμα iii :Χωρίς Backoff. Σταθμοί και AP σε κοντινοί απόσταση μεταξύ τους.



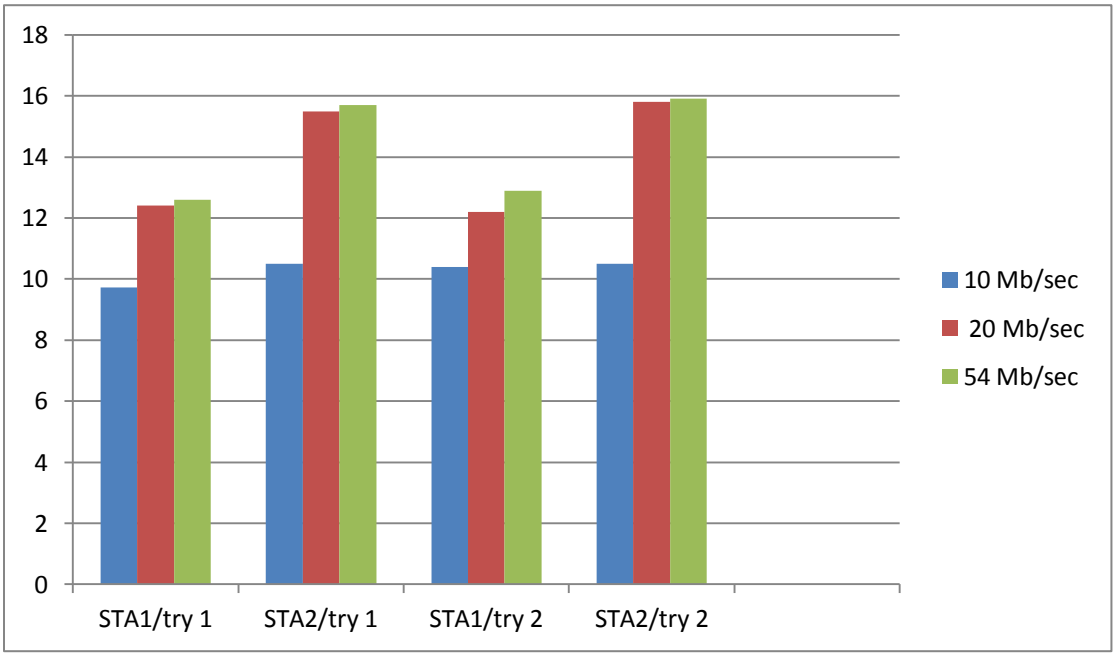
Πείραμα iv: Χωρίς Backoff. Σταθμοί σε κοντινοί απόσταση μεταξύ τους – AP σε άλλο δωμάτιο.



Πείραμα v : Soft Backoff. Σταθμοί και AP σε κοντινοί απόσταση μεταξύ τους.



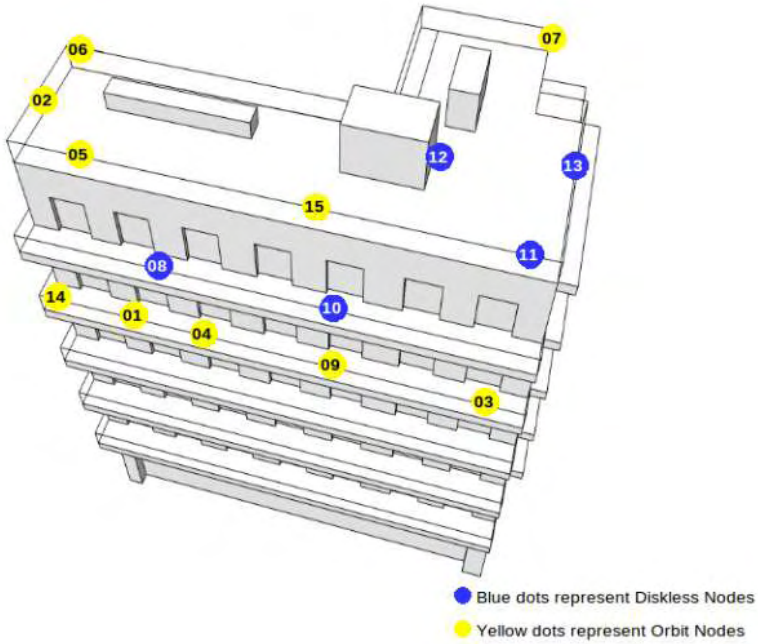
Πείραμα vi: Soft Backoff. Σταθμοί σε κοντινοί απόσταση μεταξύ τους – AP σε άλλο δωμάτιο.



Τα παραπάνω αποτελέσματα αν και δεν δείχνουν κάποια δραματική διαφοροποίηση μεταξύ των 3 τύπων backoff, επιδέχονται κάποιας αμφισβήτησης για διάφορους λόγους. Πρώτον, χρησιμοποιήθηκαν 3 διαφορετικοί υπολογιστές και 3 κάρτες διαφορετικών κατασκευαστών, δηλαδή διαφορετικές κεραίες και δυνατότητες λήψης και αποστολής. Πράγμα που φαίνεται να ευνόησε περισσότερο τον έναν από τους δυο σταθμούς. Δεύτερον, το γεγονός ότι τρέξαμε το πείραμα στα κανάλια του 80211.g εισάγει λάθη αφού υπήρξαν συχνές διαφοροποιήσεις στις συνθήκες του καναλιού από πείραμα σε πείραμα μιας και την ίδια ώρα ήταν ενεργά και άλλα AP στα κανάλια 1, 6 και 11 και εύλογα μπορούμε να υποθέσουμε ότι υπήρχε κάποια κίνηση στα δίκτυά τους κατά τη διάρκεια του πειράματος. Έτσι, δεν μπορεί κανείς να αποκλείσει την πιθανότητα κατά τη διάρκεια κάποιου από τα πειράματα η πρόσβαση στο κανάλι να ήταν πιο εύκολη ή πιο δύσκολη ευνοώντας ή δυσκολεύοντας αντίστοιχα την κατά περίπτωση έκδοση backoff που εξέταζε.
Επίσης προβληματίζει η τόσο καλή απόδοση του no Backoff. Ο λόγος που σκεφτήκαμε είναι ότι η ύπαρξη 2 μόνο σταθμών δεν δημιουργεί μεγάλο ανταγωνισμό για πρόσβαση στο κανάλι. Εξάλλου, ενδέχεται να προλάβαιναν να «συγχρονιστούν» μεταξύ τους μιας και περνούσε κάποιος χρόνος μεταξύ της έναρξης του τρεξίματος του iperf στον έναν από τον άλλον.
Γεγονός είναι ότι δεν υπάρχουν αποτελέσματα που να οδηγούν σε κάποιο πρότυπο συμπεριφοράς που να μας οδηγούν σε ασφαλή συμπεράσματα.
Για τον λόγο αυτό, πραγματοποιήσαμε μια σειρά από νέα πειράματα στο Testbed αυτήν τη φορά όπου υπήρχε η δυνατότητα να έχουμε περισσότερους των τριών κόμβους ώστε να εξετάσουμε τα 3 σχήματα σε ένα περιβάλλον με περισσότερο ανταγωνισμό.

Β) ΤΕΣΣΕΡΕΙΣ ΣΤΑΘΜΟΙ ΕΝΑ ACESS POINT

Το πείραμα διεξήχθη στις 26/09 μεταξύ 18:00 και 21:00 στο NITOS. Οι κόμβοι που συμμετείχαν στο πείραμα ήταν οι 4 (AP), 1,9,10,14 (STAs) όπως φαίνονται στην εικόνα 8.1 και στο κανάλι 36 με το πρωτόκολλο 80211.a, προς αποφυγήν παρεμβάσεων από τα δίκτυα εκτός Testbed που λειτουργούν στα 8211.b/g .



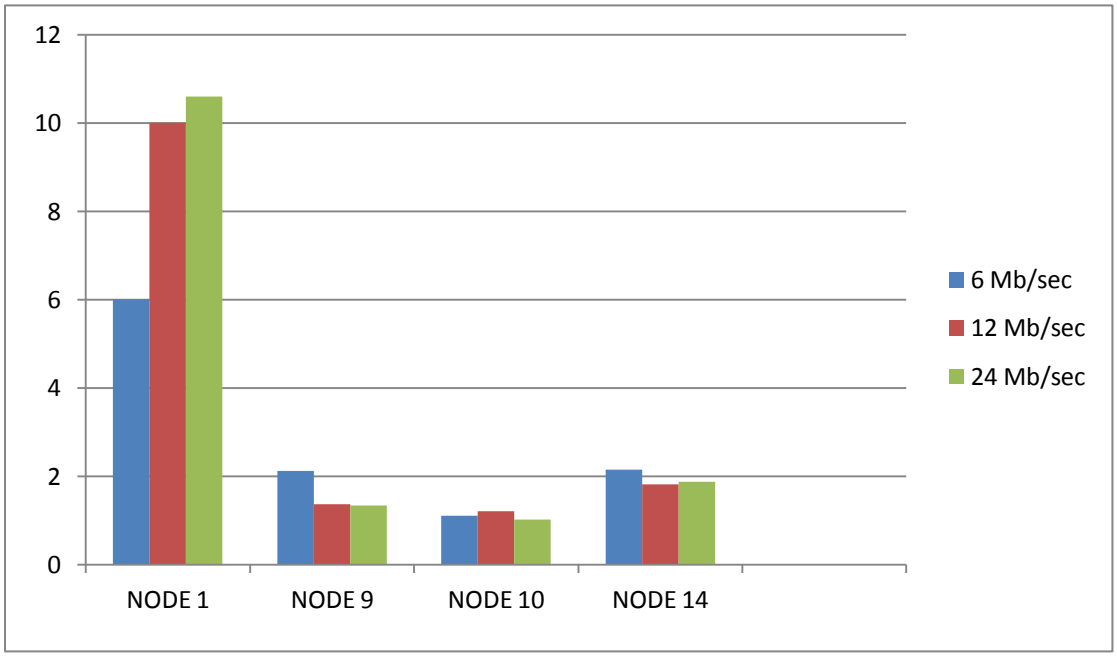
Εικόνα 8.1

Τα πειράματα έγιναν πάντα με τη χρήση του εργαλείου iperf και την αποστολή UDP πακέτων στα rates 6, 12 και 24 Mbps για διάρκεια ενός λεπτού. Το κάθε σενάριο δοκιμάστηκε τουλάχιστον 2 (a. και b.) φορές. Παρακάτω φαίνονται τα αποτελέσματα ξεχωριστά και ακολουθούν οι μέσοι όροι τους σε διαγράμματα.

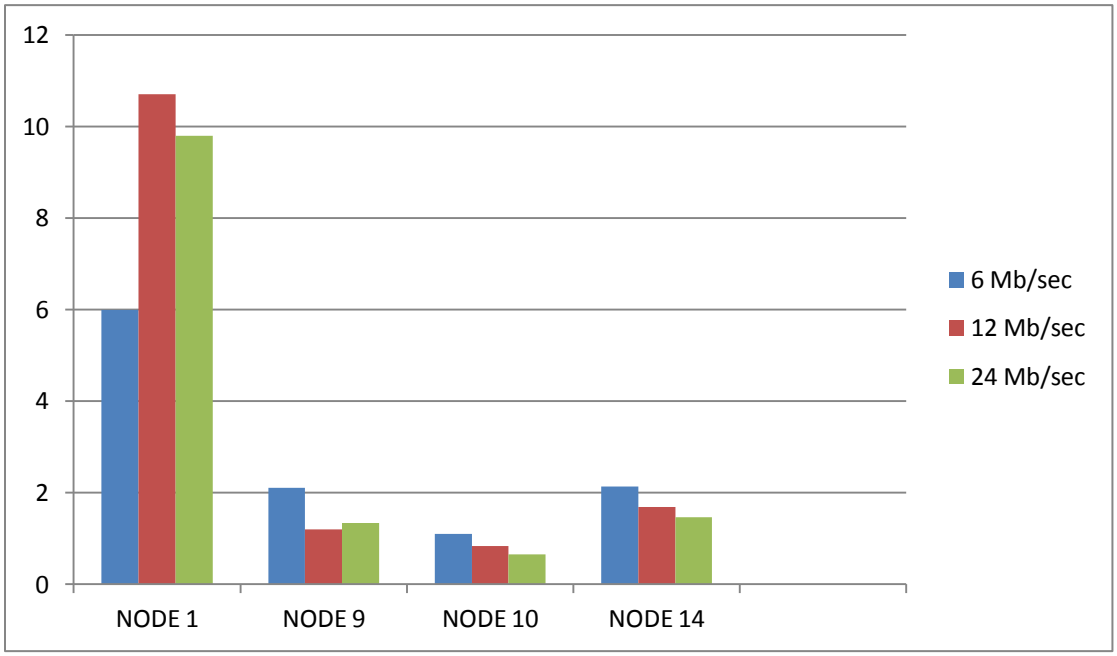
I) Ξεχωριστά αποτελέσματα:

Experiment A : Φυσικό Backoff

a.

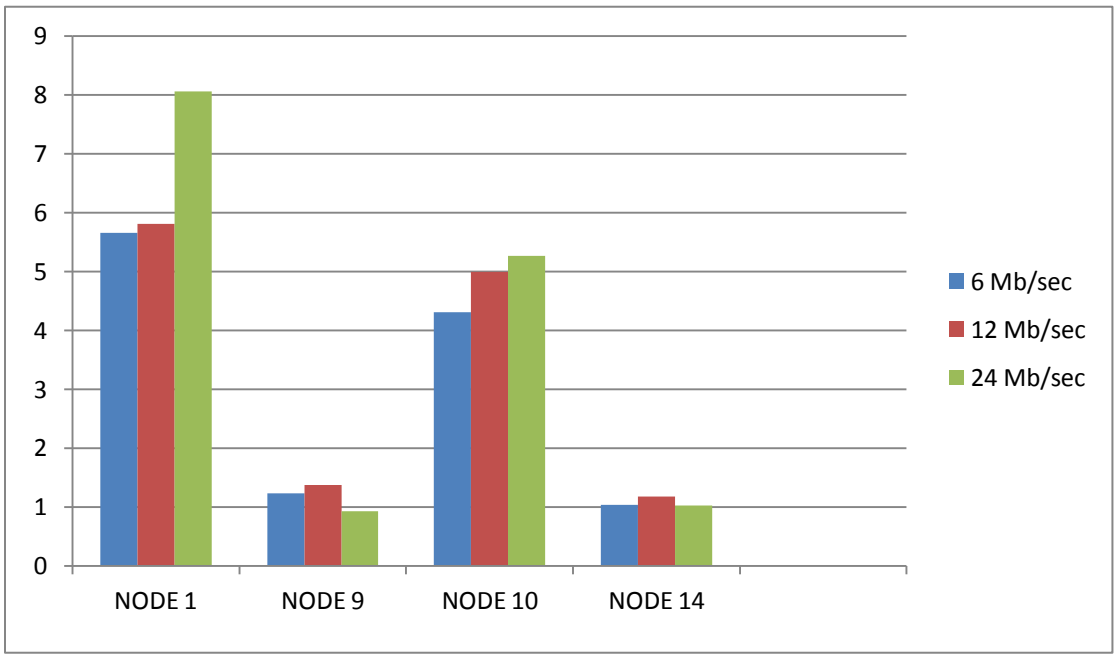


b.

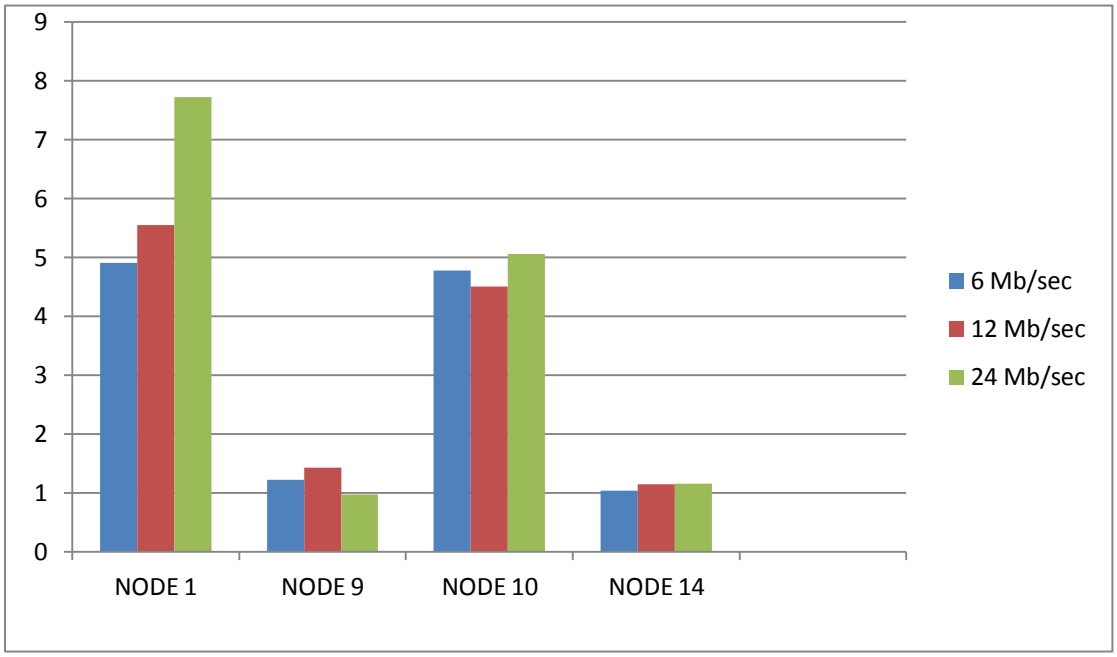


Experiment B: Soft Backoff:

a.

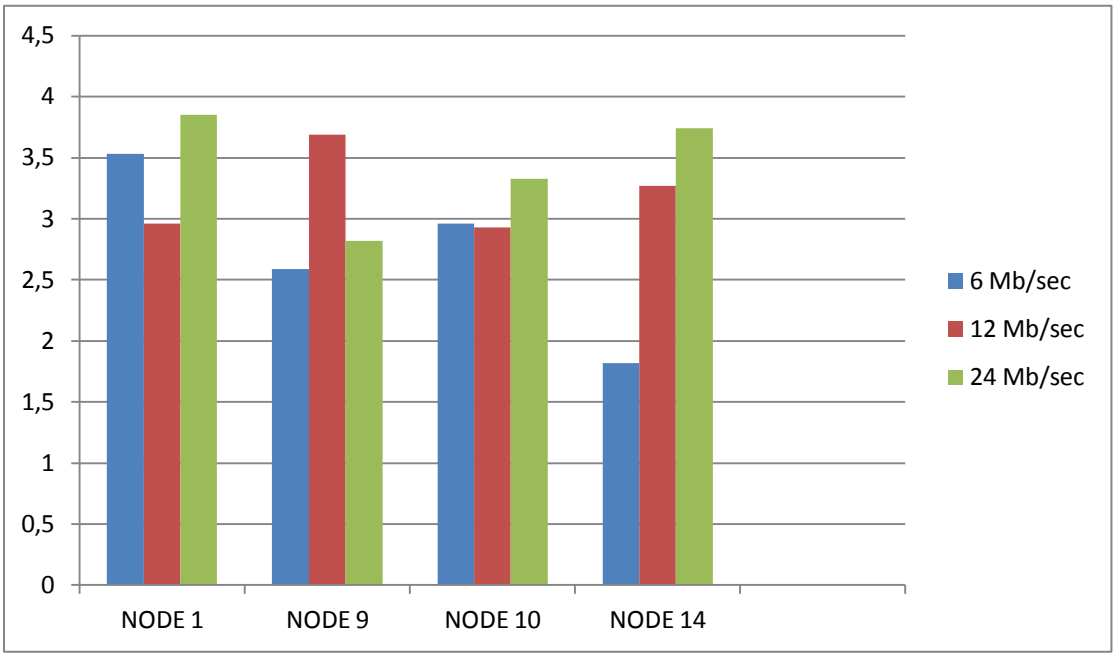


b.

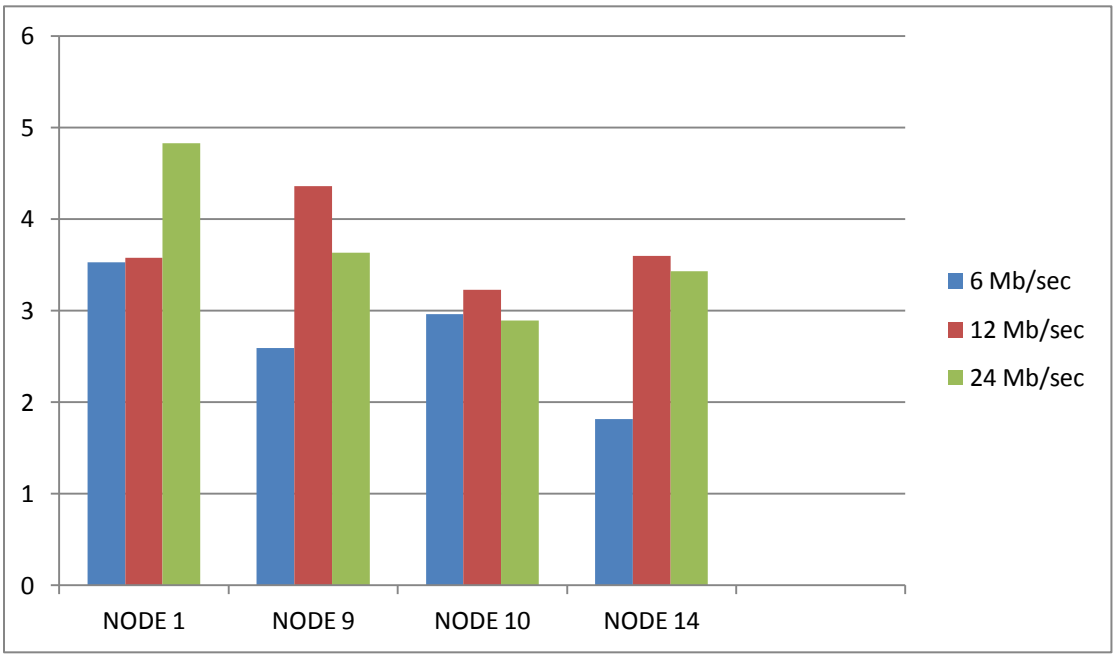


Experiment C : Χωρίς Backoff

a.

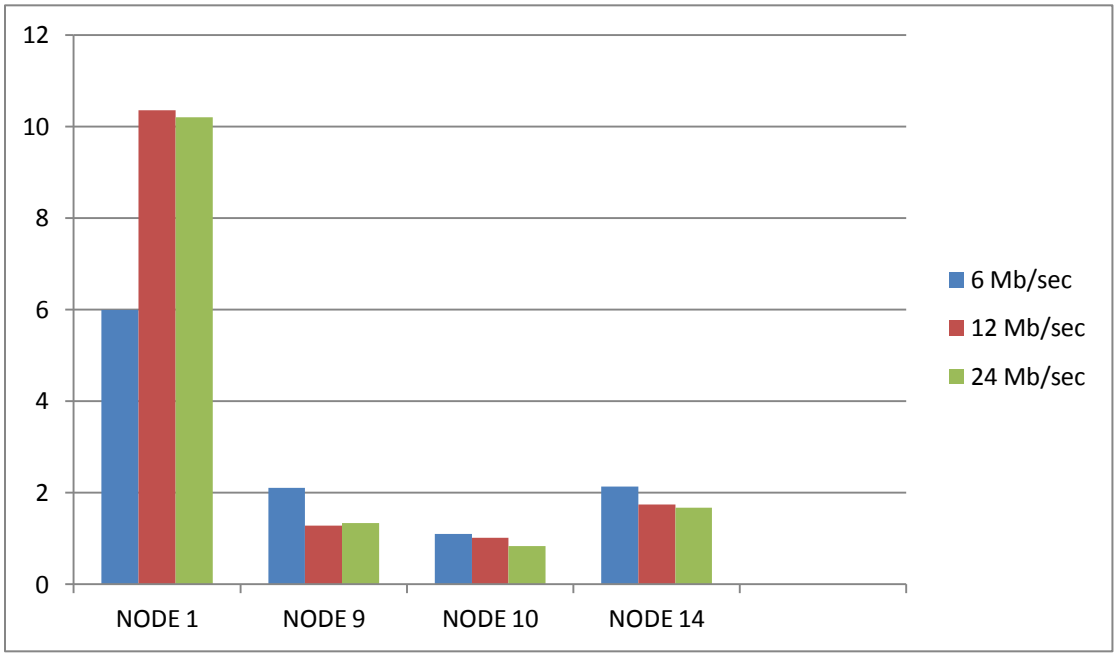


b.

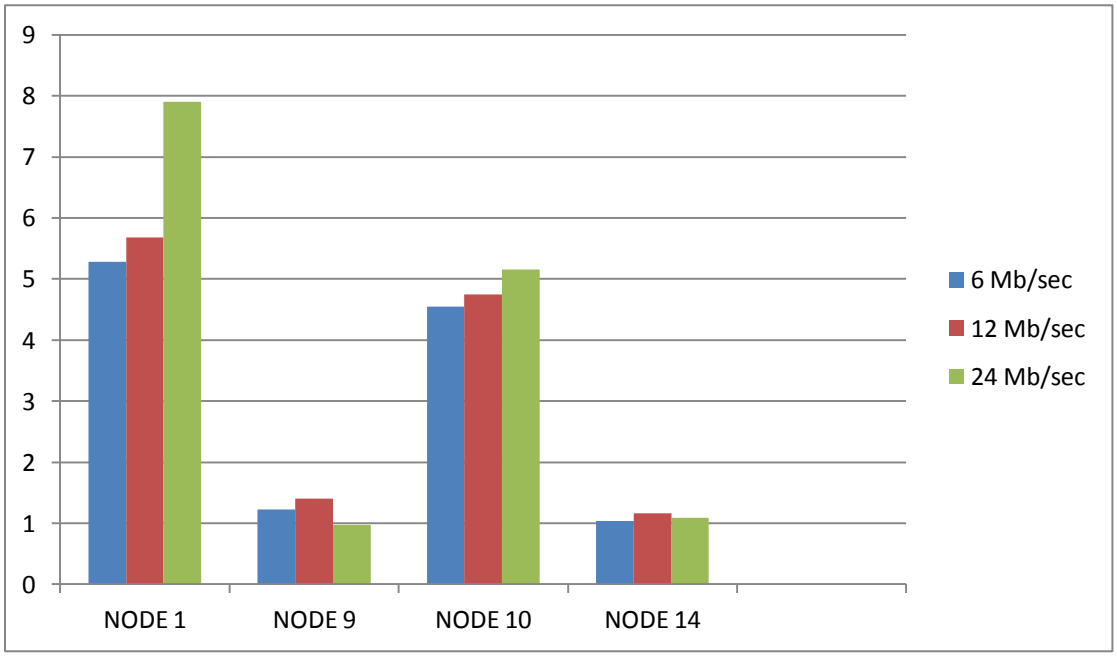


II) Μέσοι όροι των αποτελεσμάτων:

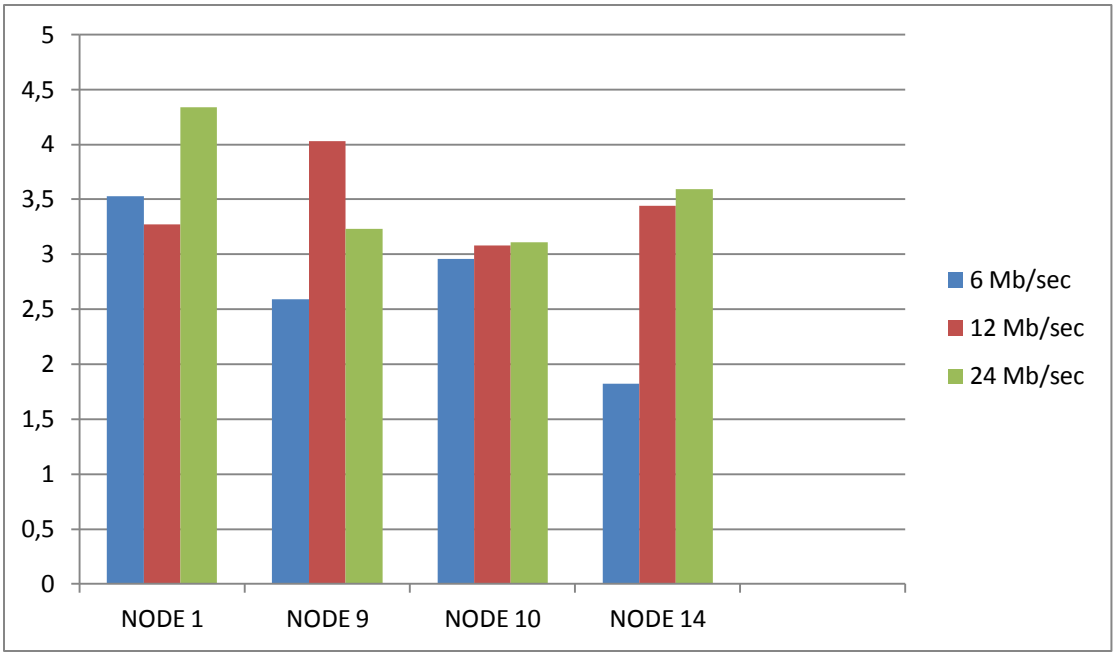
Experiment A : Physical Backoff



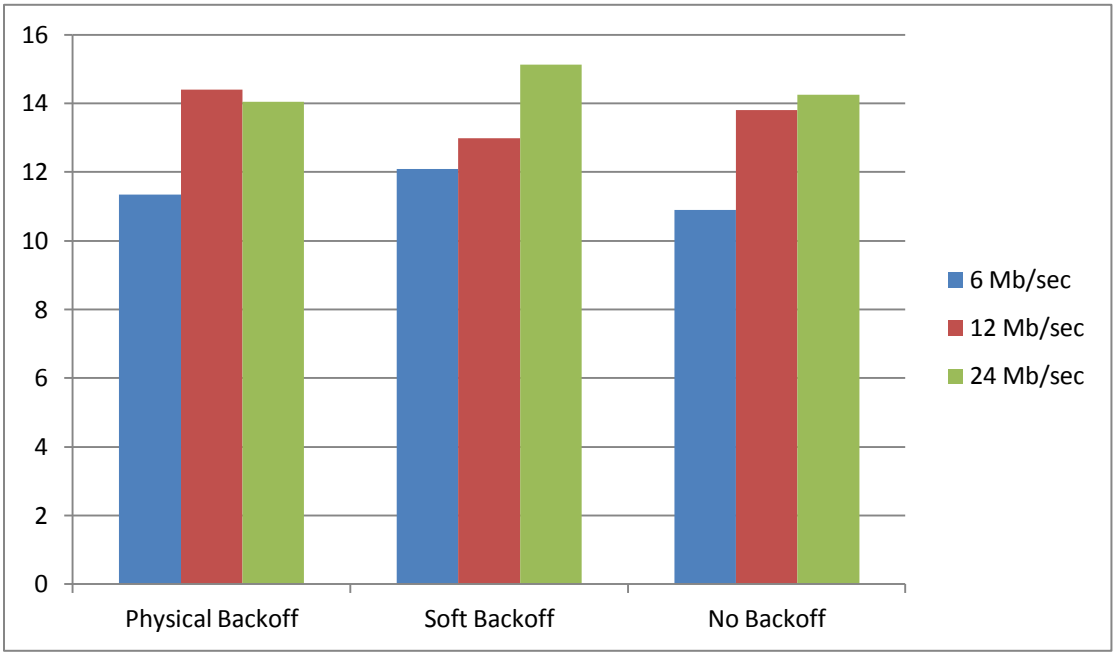
Experiment B: Soft Backoff:



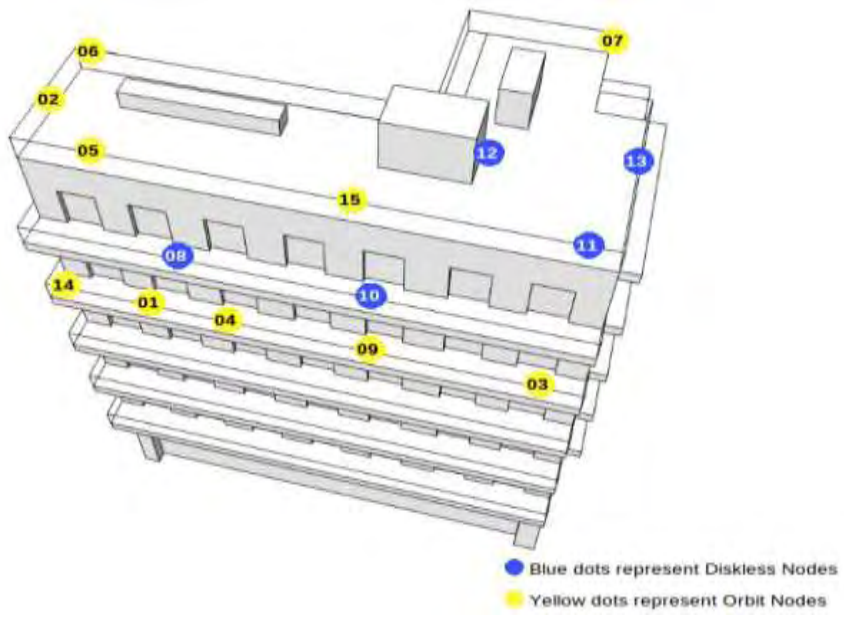
Experiment C : No Backoff



Τα παραπάνω αποτελέσματα δίνουν πολύ ενδιαφέροντα στοιχεία. Θα περίμενε κανείς το φυσικό backoff να ξεπερνάει τις άλλες δυο εκδοχές. Ιδιαίτερα όταν η μία δεν χρησιμοποιεί καθόλου Backoff!
Όμως φαίνεται το φυσικό backoff να υπολείπεται των άλλων σε ρυθμό μετάδοσης! Μάλιστα όταν προστεθούν οι ρυθμοί μετάδοσης των κόμβων έχουμε το παρακάτω διάγραμμα:



Είναι αρκετά εντυπωσιακό το γεγονός ότι το soft Backoffκαι το no Backoff ξεπερνούν το physical Backoff!
Το παραπάνω μπορεί φυσικά να είναι τυχαίο και να έχει να κάνει με τις συνθήκες του καναλιού σε κάθε πείραμα και σκαμπανεβάσματα στα links των σταθμών με το access point(κόμβος 4). Είναι γεγονός ότι ενώ τα πειράματα πραγματοποιήθηκαν στο κανάλι 36 (5,18 GHz), όταν πατούσα ανά διαστήματα «wlanconfig athx list scan» ο δείκτης S:N κάθε σταθμού άλλαζε.
Τα αποτελέσματα που βγαίνουν σίγουρα δεν οδηγούν στο συμπέρασμα ότι η καθόλου εφαρμογή του Backoff βελτιώνει την απόδοση του συστήματος. Το κρίσιμο ζήτημα είναι ο συνδυασμός του **Backoff** και του **Carrier Sensing**! Το Soft Backoff, σε κάθε σύγκρουση και συνακόλουθη επιμήκυνση του Contention window, κατά την αντίστροφη μέτρηση αγνοεί πλήρως το Carrier Sensing. Με άλλα λόγια, λόγω έλλειψης πρόσβασης στα δεδομένα του Carrier Sensing τα οποία γνωρίζει MONO το hardware και δεν φτάνουν στο driver, η αντίστροφη μέτρηση μέχρι τη μετάδοση του πακέτου γίνεται χωρίς να σταματά κάθε φορά που ανιχνεύεται κάποια δραστηριότητα στο κανάλι όπως συμβαίνει στο physical Backoff . Αυτό έχει ως συνέπεια, ανάλογα με την τοπολογία, τις συνθήκες καναλιού και την ποιότητα ζεύξης, το physical Backoff να φαίνεται ότι μειώνει το συνολικό throughput ενώ η απουσία του να την αυξάνει.
Το παραπάνω μπορεί να παρατηρηθεί αν λάβουμε υπόψιν μας την τοπολογία των κόμβων όπως φαίνονται στην παρακάτω εικόνα. Στο physical Backoff ο κόμβος με το λιγότερο throughput είναι το 10. Αυτό συμβαίνει διότι, από τη μια δεν έχει πολύ καλό link quality (συγκριτικά με τον κόμβο 1 για παράδειγμα) και βρίσκεται σε μια περιοχή ανάμεσα στους κόμβους 1 (αν και κάπως μακριά ο 1, έχει την καλύτερη ποιότητα καναλιού σε σχέση με τους



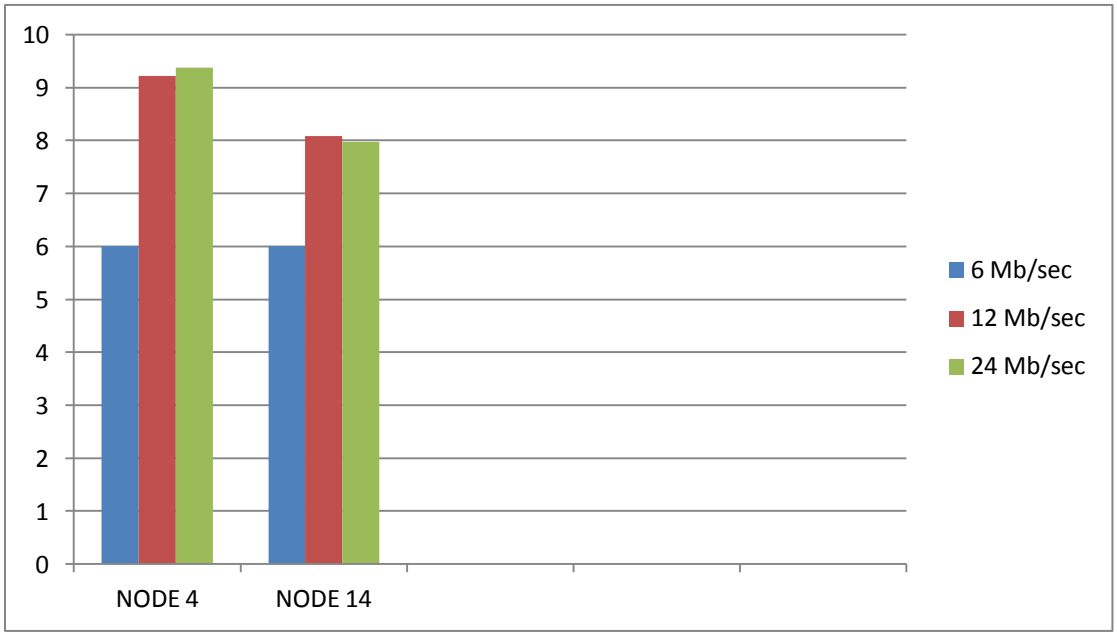
υπόλοιπους σταθμούς) και 9 οι οποίοι είναι αρκετά κοντά στο AP (4). Κάθε φορά που ο 10 θέλει να στείλει δεδομένα το CS του physical Backoff τον καθυστερεί στην αντίστροφη μέτρηση.
Ένα άλλο αξιοσημείωτο γεγονός έχει να κάνει με την εφαρμογή που χρησιμοποιήθηκε για την μέτρηση του throughput, το iperf, σε συνδυασμό με την απουσία carrier sensing. Το πρόγραμμα αυτό στέλνει άδεια πακέτα. Δεν μπορούμε να είμαστε σίγουροι ότι τα πακέτα που έφτασαν στον προορισμό είναι αναλλοίωτα και μπορούν να αξιοποιηθούν από το δέκτη σαν να επρόκειτο για πακέτα μιας πραγματικής εφαρμογής και όχι ενός testing tool. Συνεπώς μπορεί να έχουμε μεγαλύτερο throughput χωρίς το physical Backoff, δεν μπορούμε όμως να είμαστε σίγουροι ότι με το soft και το no Backoff τα πακέτα παραδίδονται ακέραια και χωρίς λάθη.
Ένα καλό και επαρκές σενάριο που θα μπορούσε να απομονώσει την επίδραση του Backoff στο throughput θα ήταν η χρήση κόμβων που έχουν ίση απόσταση μεταξύ τους και με το AP και ίδιο link quality.
Δυστυχώς εξαιτίας της τοπολογίας των κόμβων όπως φαίνεται στην παραπάνω εικόνα, τις διαφορές ισχύος , την έλλειψη διαθεσιμότητας (δεν είναι όλοι οι κόμβοι ορατοί από τους άμεσους γείτονές τους, και δεν είναι πάντα διαθέσιμοι!) και της αστάθειάς τους, η οργάνωση ενός τέτοιου πειράματος είναι αδύνατη. Υπάρχει όμως περίπτωση να πραγματοποιηθεί ένα πείραμα μ’ αυτές τις απαιτήσεις όταν αυξηθούν οι κόμβοι στους 40 όπως υπόσχεται το site του Nitlab.

C) ΔΥΟ ΣΤΑΘΜΟΙ ΕΝΑ ACESS POINT

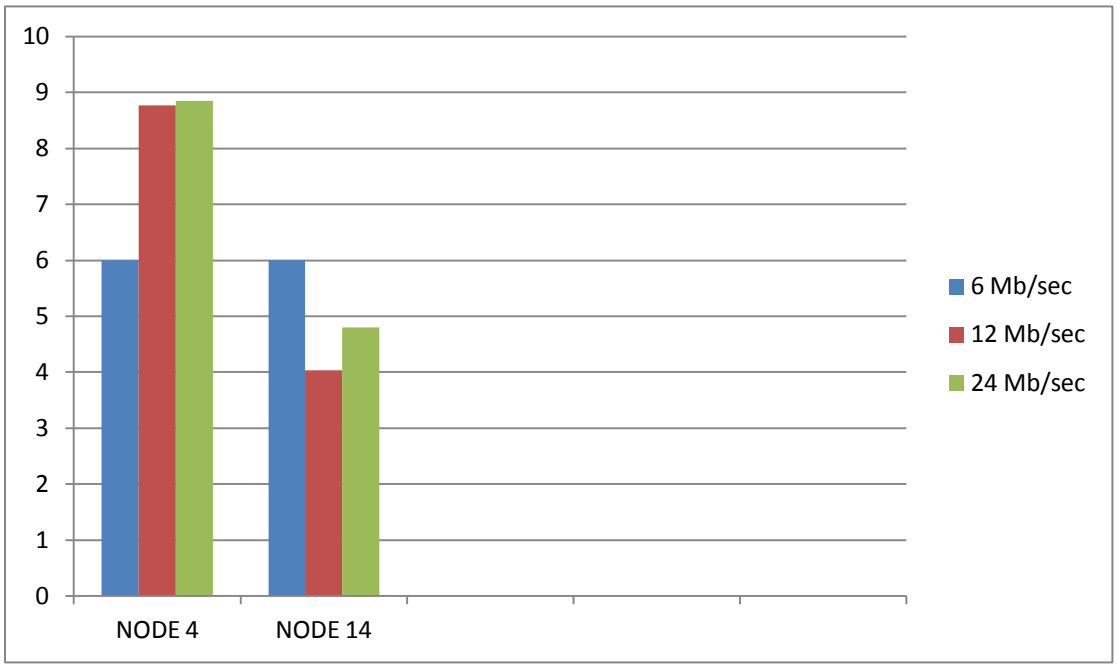
Για να προσεγγίσουμε όμως όσο το δυνατόν το παραπάνω σενάριο, έκανα και άλλο ένα πείραμα αυτήν τη φορά με 3 κόμβους μόνο. Τον 1(AP) τοποθετημένο ανάμεσα στους 4 και 14 (STAs). Δυστυχώς και πάλι υπήρχαν διαφορές S:N μεταξύ των 2 κόμβων και ο ένας είχε καλύτερη ποιότητα ζεύξης συγκριτικά με τον άλλον.
Το πείραμα αυτό, διεξήχθη μεταξύ 01:00 και 03:30 πάλι στο κανάλι 36 για τους ίδιους λόγους. Ο κόμβος 1 ήταν το AP και οι κόμβοι 4 και 14 STAs.
Όλα τα πειράματα έγιναν με τη χρήση του εργαλείου iperf και την αποστολή UDP πακέτων στα rates 6, 12 και 24 Mbps και για διάρκεια ενός λεπτού. Το κάθε σενάριο δοκιμάστηκε τουλάχιστον 2 φορές. Παρακάτω φαίνονται σε διαγράμματα τα αποτελέσματα κάθε πειράματος ξεχωριστά και ακολούθως οι μέσοι όροι τους και τέλος ο σχολιασμός.
Να σημειωθεί εδώ ότι επειδή βρήκα αποτελέσματα που μου φάνηκαν αρκετά παράξενα και με μεγάλη απόκλιση μεταξύ τους, τα πειράματα σχετικά με το Soft Backoff και physical Backoff έγιναν το καθένα από 2 φορές: Physical Backoff, Soft Backoff και Physical Backoffagain, Soft Backoffagain.

I) Ξεχωριστά Αποτελέσματα:

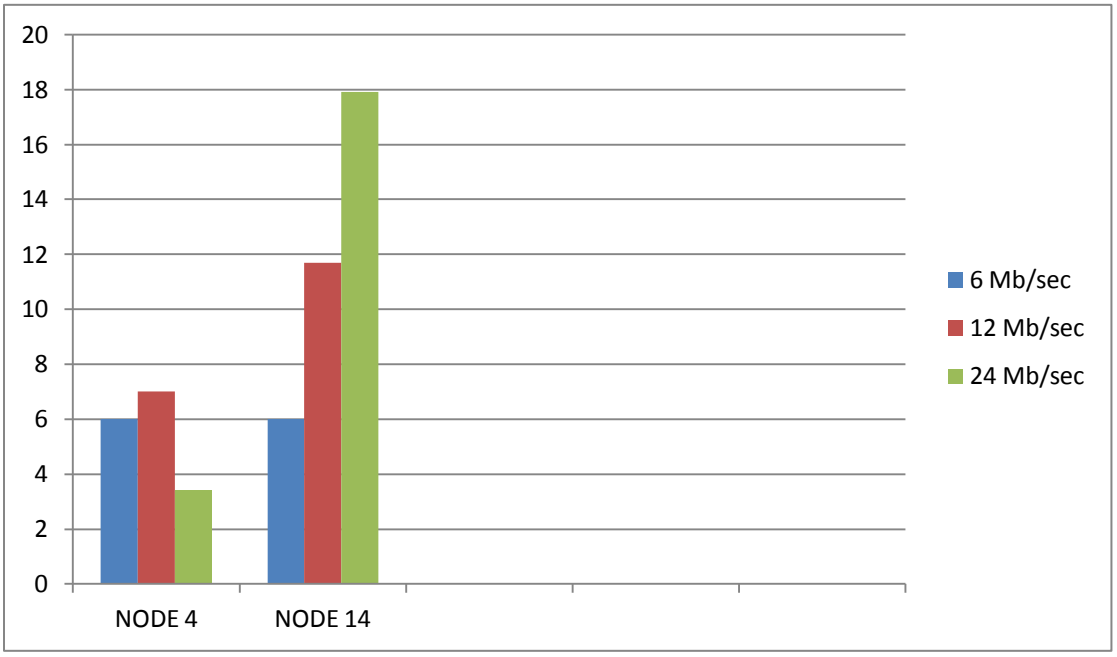
Experiment A: Soft Backoff



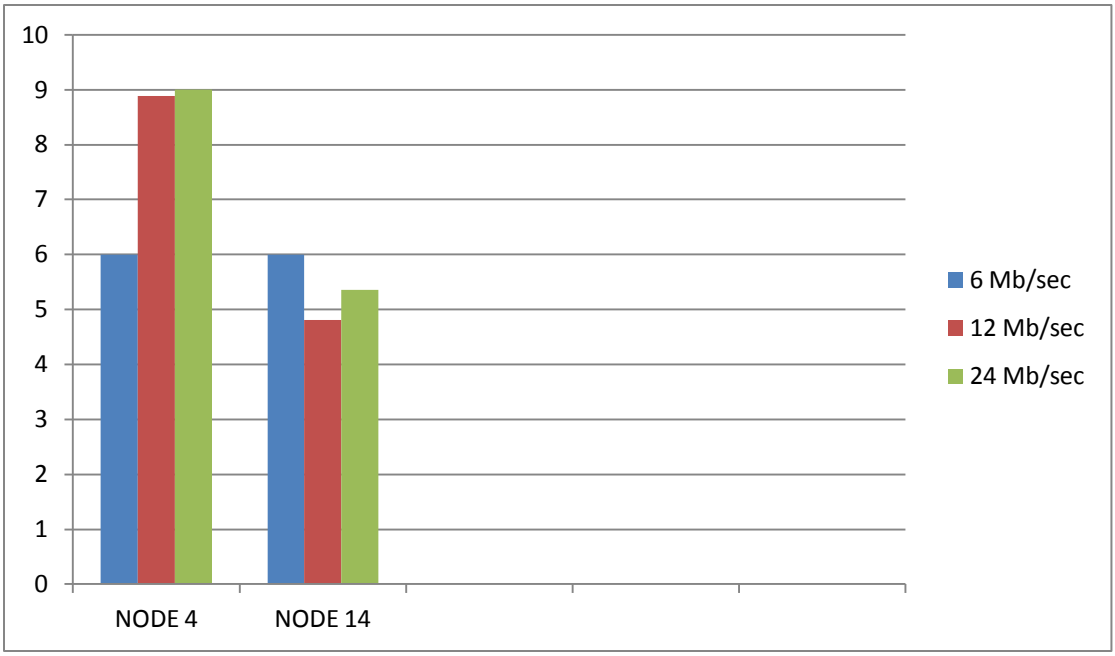
Experiment B: Physical Backoff



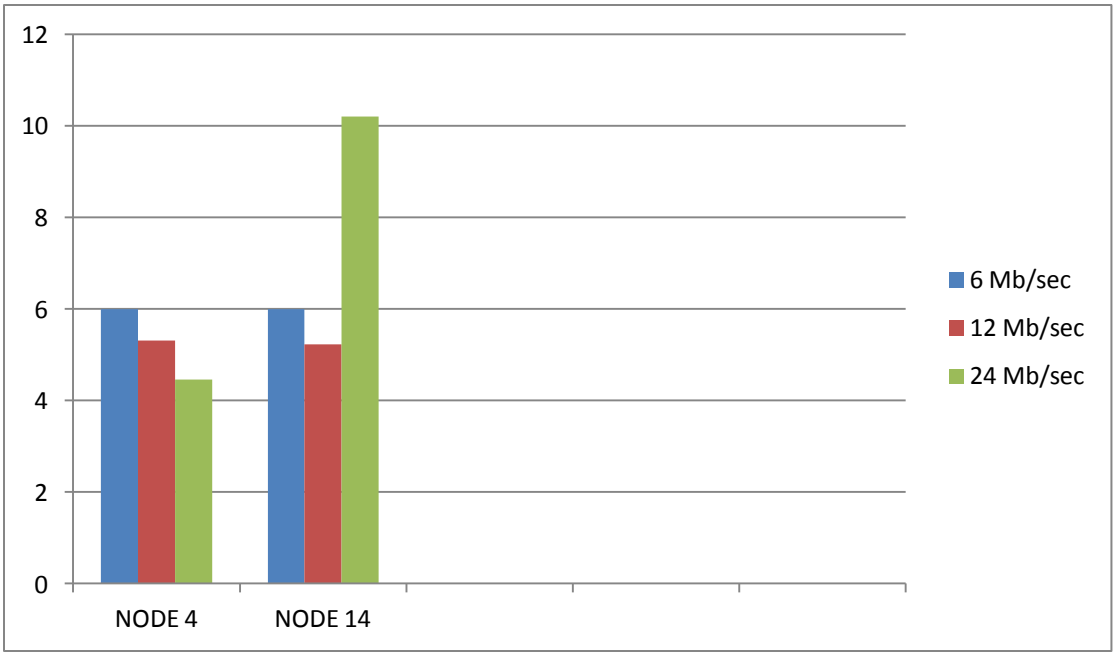
Experiment C: Soft Backoff again



Experiment D: Pysical Backoff again

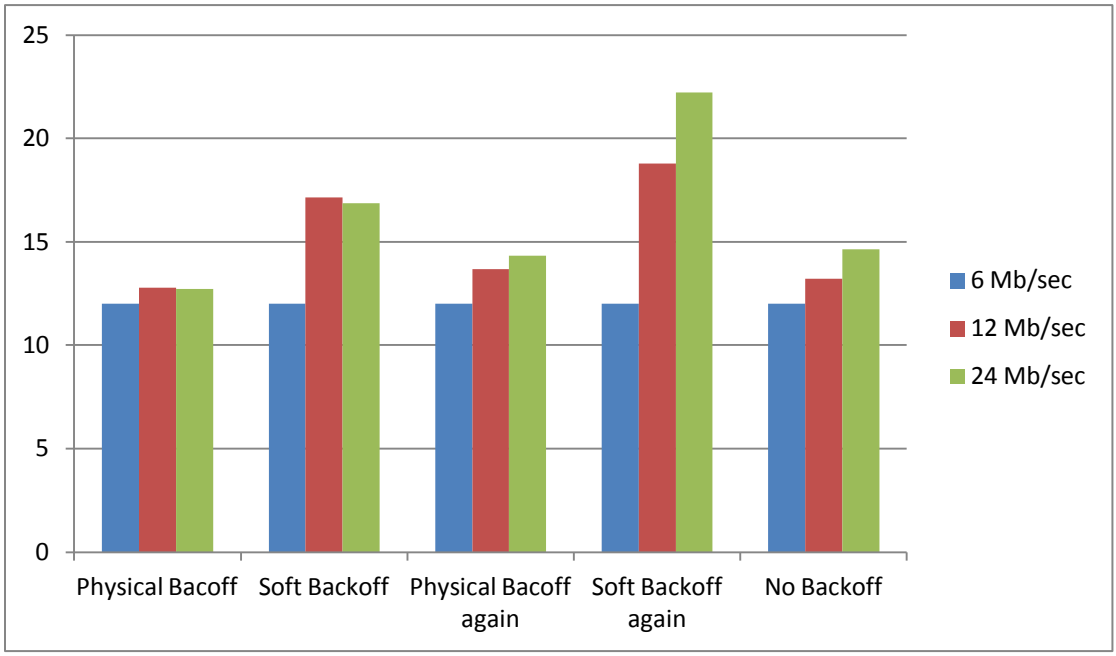


Experiment E: Χωρίς Backoff



II) Μέσοι όροι των αποτελεσμάτων

Και σ' αυτό το set πειραμάτων βλέπουμε να υπερισχύει το soft backoff! Όπως φανερώνει το παρακάτω διάγραμμα με το σύνολο των παραπάνω αποτελεσμάτων:



Το παραπάνω είναι μια ακόμα επιβεβαίωση της σημασίας του Carrier Sensing στη διαδικασία του Backoff! Και εδώ οι κόμβοι που εκπέμπουν είναι το ένα στο πεδίο ορατότητας του άλλου, μάλιστα φαίνεται ο ένας να «πνίγει» τον άλλον. Συνεπώς, το physicall backoff δε θα μετρήσει αντίστροφα αν το κανάλι είναι απασχολημένο.
Ένα άλλο αξιοσημείωτο γεγονός είναι η υπερίσχυση του soft Backoff συγκριτικά με το no Backoff! Αυτό είναι μια ένδειξη ότι ακόμα και ένα υποτυπώδες σχήμα Backoff είναι καλύτερο από το να μην υπάρχει καθόλου. Και εδώ συντρέχουν οι επιφυλάξεις σχετικά με την ακεραιότητα των πακέτων που φτάνουν στο δέκτη στα soft και no Backoff.
Να σημειωθεί τέλος, ότι η S:N ένδειξη για τη σύνδεση μεταξύ των κόμβων 14 και 1 ήταν αρκετά μεταβλητή και κυμαινόταν από 25 μέχρι 38!!!!

ΠΕΙΡΑΜΑΤΑ ΚΑΙ ΤΟ ΝΕΟ ΣΧΗΜΑ ASSOCIATION

Δυστυχώς, λόγω έλλειψης χρόνου και μη διαθεσιμότητας των κόμβων του NITOS δεν ήταν δυνατή η πειραματική μελέτη του νέου, προτεινόμενου σχήματος. Επιφυλάσσομαι όμως να δοκιμάσω το σχήμα αυτό όταν αυξηθεί ο αριθμός των κόμβων και θα είναι πιο εύκολη η ομαδοποίησή τους σε δίκτυα με πάνω από 2 μέλη και ο έλεγχος της απόδοσής τους.

ΚΕΦΑΛΑΙΟ 9

Επίλογος

Η Τεχνολογία των wireless δικτύων έχει προσφέρει πολλές ευκολίες στην καθημερινή χρήση και αξιοποίηση των υπηρεσιών δικτύου. Παράλληλα προσέφερε πολλές ευκαιρίες και προκλήσεις για έρευνα και αρκετά περιθώρια βελτιστοποίησης σε πολλές από τις διαστάσεις του.
Η διπλωματική αυτή προσπάθησε, χρησιμοποιώντας πληροφορίες που παράγονται με τη διαδικασία του Backoff αναφορικά με την ποιότητα του καναλιού, και στην κάθοδο αλλά και στην άνοδο, να προσφέρει ένα νέο σχήμα association το οποίο αντί για το rssi θα χρησιμοποιεί μια νέα μετρική λαμβάνοντας υπόψιν το μήκος του παραθύρου του Backoff, το πόσες φορές μηδενίστηκε αυτό το μήκους στη διάρκεια του χρόνου, καθώς και τον ρυθμό μετάδοσης των πακέτων.
Στην προσπάθεια αυτή αντιμετωπίστηκαν μια σειρά από δυσκολίες με μεγαλύτερη την έλλειψη ορατότητας των δεδομένων του Backoff στον driver. Για τον λόγο αυτό, απενεργοποίησα το Backoff και προσομοίωσα τη διαδικασία αυτή σε software και πραγματοποίησα μια σειρά από πειράματα τα αποτελέσματα των οποίων δίνουν ενδιαφέροντα στοιχεία σχετικά με το Backoff.
Τα αποτελέσματα της διαδικασίας του Backoff χρησιμοποιήθηκαν σε έναν αλγόριθμο ο οποίος τρέχει στο AP και συλλέγει σε γύρους δεδομένα από τους σταθμούς για να υπολογίσει την νέα μετρική. Αυτή, μεταδίδεται μαζί με το Probe Response ως απάντηση σε κάθε Probe Request που δέχεται το access point.
Για την υλοποίηση των παραπάνω, χρειάστηκαν παρεμβάσεις, προσθήκες και μικρό-αλλαγές στον κώδικα του Madwifi driver, τα οποία μπορεί ο αναγνώστης

να βρει στο τέλος στο Παράρτημα. Επίσης, για την καλύτερη εισαγωγή του αναγνώστη στο νόημα και το περιεχόμενο της παρούσας εργασίας, παρατέθηκαν επεξηγήσεις σχετικά με τις πλευρές και τμήματα των IEEE80211 δικτύων που αξιοποιούνται στην εργασία αυτή καθώς και την υλοποίησή τους στο Madwifi.
Στην διάρκεια περάτωσης αυτής της εργασίας συνάντησα κάποιες δυσκολίες οι οποίες είχαν να κάνουν κυρίως με το Madwifi. Ο driver αυτός, ενώ δίνει μεγάλες δυνατότητες παρεμβάσεων και τροποποιήσεων, όμως όπως πολλά λογισμικά ανοιχτού κώδικα είναι κακά τεκμηριωμένο . Αυτό είχε ως αποτέλεσμα, το μεγαλύτερο μέρος του χρόνου μου να έχει αναλωθεί στην εύρεση, η λέξη ανακάλυψη ίσως να ταίριαζε καλύτερα, του τρόπου λειτουργίας ακόμα και των πιο απλών διαδικασιών του driver. Και σαν να μην έφτανε αυτό, η ανάπτυξη και υποστήριξη του driver αυτού έχει σταματήσει και η ομάδα που βρισκόταν πίσω του έχει στραφεί στους ath5k και ath9k.
Παρόλα αυτά, η ενασχόληση με τον κώδικα του Madwfi, μου έδωσε σημαντικές γνώσεις για τον τρόπο δομής και τις λειτουργίες ενός driver ασύρματης κάρτας δικτύων και αξιόλογα εφόδια σχετικά με την ανάγνωση και ερμηνεία άγνωστου κώδικα, ειδικά όταν πρόκειται για modules που καλούνται από τον πυρήνα του λειτουργικού . Επίσης, ήρθα σε επαφή με την πολύ σημαντική δουλειά που γίνεται στο Nitlab πάνω στο NITOS και τις δυνατότητες που προσφέρει για έρευνα και μετρήσεις.

ΠΑΡΑΡΤΗΜΑ:

ΚΩΔΙΚΑΣ

ΚΩΔΙΚΑΣ ΓΙΑ ΤΗΝ ΠΡΟΣΟΜΟΙΩΣΗ ΤΟΥ BACKOFF:

αρχείο if_athioctl.h

```
struct ath_stats {
....
u_int32_t ast_tx_prevretry;    /* previous longretry*/
u_int32_t ast_tx_cw;       /* soft backoff contention
window*/
u_int32_t ast_tx_timesreset;    /* how many times cw
has reached 1023!*/
....
}
```

αρχείο ath/if_ath.c

```
...
#define TIME_SLOT 9
....
#define MINCW(a,b)        ((a) < (b) ? (a) : (b))
...
static struct timeval tv;
...
int soft_backoff(struct ath_softc *sc)
{
    int diff = sc->sc_stats.ast_tx_longretry - sc-
>sc_stats.ast_tx_prevretry;
    int i, cw = sc->sc_stats.ast_tx_cw;

    if((diff > 0) && (cw >= ATH_DEFAULT_CWMIN) && (cw<
ATH_DEFAULT_CWMAX)){

     for(i=0; i<diff; i++){
     cw = (cw << 1) | 1;
     }
     sc->sc_stats.ast_tx_cw =  cw =
MINCW(ATH_DEFAULT_CWMAX, cw);

     if (cw >= ATH_DEFAULT_CWMAX){
         sc->sc_stats.ast_tx_timesreset++;
     }
     sc->sc_stats.ast_tx_prevretry = sc-
>sc_stats.ast_tx_prevretry+diff;

    }

    else {cw = sc->sc_stats.ast_tx_cw =
ATH_DEFAULT_CWMIN;}

    return cw;
```

```
}



static __inline void
ath_tx_txqaddbuf(struct ath_softc *sc, struct
ieee80211_node *ni,
    struct ath_txq *txq, struct ath_buf *bf, int
framelen)
{
....
unsigned long rand_int,wt;
....
get_random_bytes(&rand_int, sizeof (unsigned long));
     do_gettimeofday(&tv);
     printk("ath_addbuf: microsecs before delay:
%6ld\n", tv.tv_usec);
     wt = (rand_int % (soft_backoff(sc)*TIME_SLOT));
//waiting time
     udelay(wt);
     do_gettimeofday(&tv);
     printk("ath_addbuf: microsecs after delay:
%6ld\n", tv.tv_usec);
....

}

static struct ath_txq *
ath_txq_setup(struct ath_softc *sc, int qtype, int
subtype)
{
...
//Disabling Backoff
qi.tqi_qflags |= HAL_TXQ_BACKOFF_DISABLE;
```

```
...
}
```

ΚΩΔΙΚΑΣ ΓΙΑ ΤΟ ΝΕΟ ΣΧΗΜΑ ASSOCIATION:

ΣΤΟ ACCESS POINT:

αρχείο net80211/ieee80211.h

```
...
#define IEEE80211_FC0_SUBTYPE_ROUND_RESP    0x60
...
struct ieee80211_ie_country {
...
u_int32_t country_mymetric;    /* my metric!*/
...
}
...
```

αρχείο net80211/ieee80211_linux.h

```
...
#define    le64toh(_x)    le64_to_cpu(_x)
```

```
#define    htole64(_x)    cpu_to_le64(_x)

#define    be64toh(_x)    be64_to_cpu(_x)

#define    htobe64(_x)    cpu_to_be64(_x)

...
```

αρχείο net80211/ieee80211_var.h

```
...
struct ieee80211com {
...
    u_int32_t            ic_round;
    u_int32_t            ic_myround;
    u_int32_t            ic_resets;
    u_int32_t            ic_cw;
    u_int32_t            ic_mymetric;
...
}
...
extern u_int8_t whoami[IEEE80211_ADDR_LEN];
extern struct timer_list send_round_req;
u_int32_t calc_mymetric(struct ieee80211_node *);
...
```

αρχείο net80211/ieee80211_node.h

```
...
#define MAX_NODES 20

struct my_node {
    struct ieee80211_node *my_ni;
    u_int8_t     mynd_macaddr[IEEE80211_ADDR_LEN];
/* nodes mac address*/
    u_int8_t     mynd_bssid[IEEE80211_ADDR_LEN];
/* nodes mac address*/
    u_int32_t    mynd_round;                /* current
round */
```

```
    u_int32_t    mynd_roundsin;                 /*
node's rounds in game*/
    u_int32_t    mynd_resets;                /* nodes
CW resets*/
    u_int32_t    mynd_cw;
    u_int8_t    not_empty;                /* 1 when
my_node_table[i] is occupied*/
};

extern struct my_node my_node_table[MAX_NODES];
int node_check(u_int8_t *mnaddr);
int mynode_find(u_int8_t *mnaddr);
void mynode_print(u_int8_t *mnaddr);
...
```

αρχείο net80211/ieee80211_beacon.c

```
...
static int bcntr = 0;

#define ROP    10;
#define USIZE    32;
#define LOP    22;
u_int32_t int_part(u_int32_t metric){
    u_int32_t temp1 = metric;

    temp1 >>= ROP;

return temp1;
}

u_int32_t double_part(u_int32_t metric){

    u_int32_t temp1, temp2;
    temp1 = metric;
    temp1 <<= LOP;
    temp2 = temp1 >> LOP;
return temp2;
}
...
int
```

```
ieee80211_beacon_update(struct ieee80211_node *ni,
    struct ieee80211_beacon_offsets *bo, struct sk_buff
*skb,
    int mcast, int *is_dtim)
{
...
struct ieee80211_frame *wh;
...
bcntr++;
wh = (struct ieee80211_frame *)skb->data;
    if((bcntr%35)==0){
    ic->ic_round++;
    ic->ic_mymetric = calc_mymetric(ni);
    wh->i_dur = 1;
    bcntr=0;

    }

...


}
```

αρχείο net80211/ieee80211_node.c

```
...
#define    IEEE80211_RATE2MBS(r)    (((r) &
IEEE80211_RATE_VAL) / 2)
struct my_node my_node_table[MAX_NODES];
...
#define ROP    10;
#define USIZE    32;
#define LOP    22;

u_int32_t calc_mymetric(struct ieee80211_node *ni){

u_int32_t upl, downl, temp1, temp2, temp3;
u_int8_t nrate = 1;
u_int8_t srate =1;
int i;
```

```
// calculating downlink metric:
    temp1 = ni->ni_ic->ic_resets + ni->ni_ic->ic_cw;
    temp1 <<= ROP;
if (ni->ni_rates.rs_rates[ni->ni_txrate] >0) nrate =
IEEE80211_RATE2MBS(ni->ni_rates.rs_rates[ni-
>ni_txrate]);
    downl = temp1 / nrate;

// calculating uplink metric:
temp2 = temp3 = upl = 0;

     for(i = 0; i < MAX_NODES; i++){
        if(my_node_table[i].not_empty == 0) {
            break;
        }
        temp2 = my_node_table[i].mynd_resets +
my_node_table[i].mynd_cw;
        temp2 <<= ROP;
        if (my_node_table[i].mynd_round>0) srate =
IEEE80211_RATE2MBS(my_node_table[i].mynd_round);
        upl += ((ni->ni_ic->ic_round -
my_node_table[i].mynd_roundsin)*temp2) / srate;
my_node_table[i].mynd_resets +
my_node_table[i].mynd_cw: %u, after shift ROP: %u \n",
temp3, my_node_table[i].mynd_resets +
my_node_table[i].mynd_cw, temp2 );
    }

return (downl + upl);

}
...
int node_check(u_int8_t *mnaddr)
{
    int i, val = 0;
    for(i = 0; i < MAX_NODES; i++){

 if(IEEE80211_ADDR_EQ(my_node_table[i].mynd_macaddr,
mnaddr)){
            val = 1;
            break;
```

```
        }

    }

    if(i > MAX_NODES) val = -1; //reached the end!
    return val;
}

//returns the next empty slot in node_table:
int mynode_find_empty(int isnt) //fake argument, just
to keep compiler happy
{
    int i;
    int val;
    for(i = isnt; i < MAX_NODES; i++){
        if(my_node_table[i].not_empty == 0){
            val = i;
            break;
        }
    }

    if(i >= MAX_NODES) val = -1;

    return val;
}

//returns the node's position in my_node_table:
int mynode_find(u_int8_t *mnaddr)
{
    int i;
    int val;
    for(i = 0; i < MAX_NODES; i++){

if(IEEE80211_ADDR_EQ(my_node_table[i].mynd_macaddr,
mnaddr)){
            val = i;
            break;
        }
    }
    if(i >= MAX_NODES) val = -1;
    return val;
```

```
}
void mynode_print(u_int8_t *mnaddr)
{
    int i;
    for(i = 0; i < MAX_NODES; i++){
        if(my_node_table[i].not_empty ==
0)break;//there is no node!

if(IEEE80211_ADDR_EQ(my_node_table[i].mynd_macaddr,
mnaddr)){
            printk("node's position in nodes table:%d
\n", i);
            printk("node's mac address: ["MAC_FMT"]\n",
MAC_ADDR(my_node_table[i].mynd_macaddr));
            printk("node's bssid: ["MAC_FMT"]\n",
MAC_ADDR(my_node_table[i].mynd_bssid));
            printk("round/rate: %u\n",
IEEE80211_RATE2MBS(my_node_table[i].mynd_round));
            printk("node's rounds:%u \n",
my_node_table[i].mynd_roundsin);
            printk("node's resets:%u \n",
my_node_table[i].mynd_resets);
            printk("node's cw:%u \n",
my_node_table[i].mynd_cw);
            printk("node's not_empty:%u \n",
my_node_table[i].not_empty);
            //break;
        }
    }

    return;
}

ieee80211_node_join(struct ieee80211_node *ni, int
resp)
{
...
//node allocation
        isnode_in_table = node_check(ni->ni_macaddr);
        // check if there is already such a node in
table + if it associated=> has the same ibssid with my
```

```
macaddress
        if((isnode_in_table == 0) &&
IEEE80211_ADDR_EQ(whoami, ni->ni_bssid)){
            mynt_entry = mynode_find_empty(0);
            if(mynt_entry > -1){

IEEE80211_ADDR_COPY(my_node_table[mynt_entry].mynd_maca
ddr, ni->ni_macaddr);

IEEE80211_ADDR_COPY(my_node_table[mynt_entry].mynd_bssi
d, ni->ni_bssid);
                my_node_table[mynt_entry].not_empty =
1;

//memcpy(my_node_table[mynt_entry].my_ni, ni, sizeof
(ni));
                my_node_table[mynt_entry].my_ni = ni;
                printk("Lets see what's in ni! ni= %p
my_ni = %p \n ", ni, my_node_table[mynt_entry].my_ni);
                printk("ieee80211_node.c line: 2110,
node_join from ["MAC_FMT"] and
my_node_table[mynt_entry].mynd_macaddr =
["MAC_FMT"]..\n", MAC_ADDR(ni-
>ni_macaddr),MAC_ADDR(my_node_table[mynt_entry].mynd_ma
caddr));
            }
            else isnode_in_table = mynt_entry;
        }
//handling just MAX_NODES = 20 nodes!
        else if (isnode_in_table == -1){
            IEEE80211_SEND_MGMT(ni, resp,
            IEEE80211_REASON_ASSOC_TOOMANY);
            ieee80211_node_leave(ni);
            printk("ieee80211_node.c line: 2119,
node_join from else block isnode_in_table ==
%d..\n",isnode_in_table);
            return;
        }

        else printk("ieee80211_node.c line: 2123,
```

```
node_join  isnode_in_table = %d.. so it's already
there!\n",isnode_in_table);

....
}
```

αρχείο net80211/ieee80211_input.c

```
...
//my address
u_int8_t whoami[IEEE80211_ADDR_LEN];
...
int
ieee80211_recv_mgmt(struct ieee80211vap *vap,
    struct ieee80211_node *ni_or_null, struct sk_buff
*skb,
    int subtype, int rssi, u_int64_t rtsf)
{
...
int mynt_entry;
...
switch (subtype) {

    case IEEE80211_FC0_SUBTYPE_ROUND_RESP:

        /*
         * Round Response frame format
         *    [4] round
         *    [4] rounds in game
         *    [4] resets
         *    [4] CW
        */

mynt_entry = mynode_find(wh->i_addr2);

        if(mynt_entry!=-1){

        my_node_table[mynt_entry].mynd_round =
le32toh(*(__le32 *)frm); //RATE
```

```
        frm += 4;
        my_node_table[mynt_entry].mynd_roundsin =
le32toh(*(__le32 *)frm);
        frm += 4;
        my_node_table[mynt_entry].mynd_resets =
le32toh(*(__le32 *)frm);
        frm += 4;
        my_node_table[mynt_entry].mynd_cw =
le32toh(*(__le32 *)frm);

mynode_print(wh->i_addr2);
        }
        else {
            printk("entry not found in node
table=>DISCARD!\n");
            IEEE80211_DISCARD(vap, IEEE80211_MSG_ANY,
wh, "mgt",
            "subtype 0x%x not handled", subtype);
            vap->iv_stats.is_rx_badsubtype++;

            break;
        }

case IEEE80211_FC0_SUBTYPE_ASSOC_REQ:
    case IEEE80211_FC0_SUBTYPE_REASSOC_REQ: {
    ...
    IEEE80211_ADDR_COPY(whoami, vap->iv_myaddr);
    ...
}
...

}//switch
...
}//recv_mgmt
```

αρχείο net80211/ieee80211_output.c

```
...
u_int8_t *
ieee80211_add_country(u_int8_t *frm, struct
ieee80211com *ic)
```

```
{
//adding my metric
    memcpy(&ic->ic_country_ie.country_mymetric, &ic-
>ic_mymetric, sizeof(u_int32_t));
    /* add country code */
    memcpy(frm, (u_int8_t *)&ic->ic_country_ie,
        ic->ic_country_ie.country_len + 2);
    frm +=  ic->ic_country_ie.country_len + 2;
    return frm;
}
...
ieee80211_send_mgmt(struct ieee80211_node *ni, int
type, int arg)
{
...
switch (type) {
    case IEEE80211_FC0_SUBTYPE_PROBE_RESP:
    ...
    /* country code */
        if ((ic->ic_flags & IEEE80211_F_DOTH) ||
            (ic->ic_flags_ext &
IEEE80211_FEXT_COUNTRYIE))
            frm = ieee80211_add_country(frm, ic);
    ...
}
...
}
```

ΣΤΟ STATION:

αρχείο ath/if_athioctl.h

```
...
u_int32_t ast_tx_myrounds;    /* how many rounds i 'm
```

```
in the "game"*/
...
```

αρχείο net8211/ieee80211.h

```
...
#define IEEE80211_FC0_SUBTYPE_ROUND_RESP    0x60
...
```

αρχείο net8211/ieee80211_linux.h

```
...
#define    le64toh(_x)    le64_to_cpu(_x)
#define    htole64(_x)    cpu_to_le64(_x)
#define    be64toh(_x)    be64_to_cpu(_x)
#define    htobe64(_x)    cpu_to_be64(_x)
...
```

αρχείο net8211/ieee80211_linux.h

```
...
struct ieee80211com {
...
    u_int32_t            ic_rates;
    u_int32_t            ic_myround;
    u_int32_t            ic_resets;
    u_int32_t            ic_cw;
    u_int32_t            ic_mymetric;
...
}
```

αρχείο net8211/ieee80211_input.c

```
...
static u_int8_t myap[IEEE80211_ADDR_LEN];
...
#define ROP 10
#define LOP 22

u_int32_t int_part(u_int32_t metric){
```

```
u_int32_t temp1 = metric;

    temp1 >>= ROP;

return temp1;
}

u_int32_t double_part(u_int32_t metric){

u_int32_t temp1, temp2;
    temp1 = metric;
    temp1 <<= LOP;
    temp2 = temp1 >> LOP;
return temp2;
}
...
ieee80211_recv_mgmt(struct ieee80211vap *vap,
    struct ieee80211_node *ni_or_null, struct sk_buff
*skb,
    int subtype, int rssi, u_int64_t rtsf)
{
...
u_int32_t my_metric;
struct ieee80211_ie_country *cntry;
...
switch (subtype) {
    case IEEE80211_FC0_SUBTYPE_PROBE_RESP:
    case IEEE80211_FC0_SUBTYPE_BEACON: {
...
if((wh->i_dur == 1) &&(IEEE80211_ADDR_EQ(myap, ni-
>ni_macaddr))){
                IEEE80211_SEND_MGMT(ni,
IEEE80211_FC0_SUBTYPE_ROUND_RESP, 0);
            }
            else if (wh->i_dur > 1)
printk("ieee80211_input.c: wh->i_dur = %u, when dur >
```

```
1, that's when i get my metric!  \n", wh->i_dur);

...
switch (*frm) {
...
case IEEE80211_ELEMID_COUNTRY:
                scan.country = frm;

cntry = (struct ieee80211_ie_country *)scan.country;
                        memcpy(&my_metric, &cntry-
>country_mymetric, sizeof(u_int32_t)); // maybe a '='
would suffice
}
...
IEEE80211_ADDR_COPY(myap, ni->ni_macaddr);
break;
...
}
...
}
```

αρχείο net8211/ieee80211_output.c

```
...
#include <ath/if_athioctl.h>
#include <ath/if_athvar.h>
#define IEEE80211_RATE2MBS(r)    (((r) &
IEEE80211_RATE_VAL) / 2)
...
struct sk_buff *
ieee80211_encap(struct ieee80211_node *ni, struct
sk_buff *skb, int *framecnt)
{

ic->ic_rate = ni->ni_rates.rs_rates[ni->ni_txrate];
...
```

```
}
...
int
ieee80211_send_mgmt(struct ieee80211_node *ni, int
type, int arg)
{
...
switch (type){
...
case IEEE80211_FC0_SUBTYPE_ROUND_RESP:
        /*IEEE80211_NOTE(vap, IEEE80211_MSG_ASSOC, ni,
            "send station disassociate (reason %d)",
arg);*/

        /*
         * Round Response frame format
         *    [4] rate
         *    [4] rounds in game
         *    [4] resets
         *    [4] CW
        */
        skb = ieee80211_getmgtframe(&frm,
sizeof(u_int32_t)
                + sizeof(u_int32_t) + sizeof(u_int32_t)
                + sizeof(u_int32_t));

        if (skb == NULL)
            senderr(ENOMEM, is_tx_nobuf);

        /* rounds */
        *(__le32 *)frm = htole32(ic->ic_rounds);
        frm += 4;

        /* my rounds in game*/
        *(__le32 *)frm = htole32(sc-
>sc_stats.ast_tx_myrounds++);
```

```
        frm += 4;

        /* resets */
        *(__le32 *)frm = htole32(sc-
>sc_stats.ast_tx_timesreset);
        frm += 4;

        /* CW */
        *(__le32 *)frm = htole32(sc-
>sc_stats.ast_tx_cw);

        break;
...
}
...
}
```